DSMC PRESS
TECHNICAL REPORT, TR 4-94

DEFENSE SYSTEMS MANAGEMENT COLLEGE
FORT BELVOIR, VIRGINIA

AN ABSTRACT MODEL OF ROGUE CODE INSERTION INTO RADIO FREQUENCY WIRELESS NETWORKS

THE EFFECTS OF COMPUTER VIRUSES ON THE PROGRAM MANAGEMENT OFICE

LTC CHRISTOPHER V. FEUDO
PROFESSOR OF ENGINEERING, DSMC

A Dissertation
Presented to the
Faculty of
The School of Engineering and
Computer Science
The George Washington University

APRIL 1994

PREFACE

This research examines the effects of computer viruses* to the PMO. Computer viruses continue to be a real threat to all computing systems, to include traditional and wireless based networks. We will examine ways of mitigating this new threat. Trends in increased computer, operating system, and network standardization, as well as increased use of distributed systems, and computer connectivity enhance this viability of attacking targeted hosts via radio frequency.

Hardware acquisition managers, like software acquisition managers (Dobbins, 1994)[1], must follow basic rules. Program managers must recognize that both hardware and software issues, just as software issues alone, can kill them.

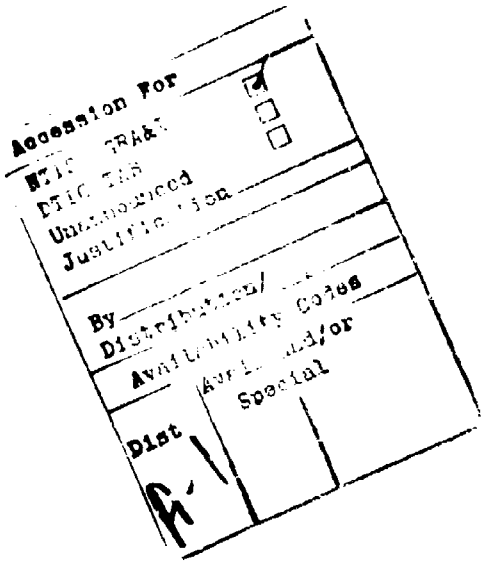
The rapidly growing popularity of wireless LANs is proliferating intruders' opportunities to infect computing systems via radio frequency (RF). Telecommunication hardware and software components, each component's specifications, and the technology to inject computer viruses via RF communication channels are proven and readily available. Unauthorized users can purchase "telecommunications saturday night specials" at

---

*Defined as a "rogue program" throughout the dissertation to denote any type of malicious code, such as logic and time bombs, worms, and trojan horses

many electronics outlets to insert surreptitious code into RF communication channels.

The implications for the PMO are mind-boggling - aircrafts, weapon systems, "smart" bomb technology, and C^3 face an additional insidious threat, which may gravely affect the security of the United States.



ABSTRACT

This dissertation demonstrates that inadequately protected wireless LANs are more vulnerable to rogue program attack than traditional LANs. Wireless LANs not only run the same risks as traditional LANs, but they also run additional risks associated with an open transmission medium. Intruders can scan radio waves and, given enough time and resources, intercept, analyze, decipher, and reinsert data into the transmission medium.

This dissertation describes the development and instantiation of an abstract model of the rogue code insertion process into a DOS-based wireless communications system using Radio Frequency (RF) atmospheric signal transmission. The model is general enough to be applied to widely used target environments such as UNIX, Macintosh and DOS operating systems. The methodology and three modules, the prober, activator, and trigger modules, to generate rogue code and insert it into a wireless LAN were developed to illustrate the efficacy of the model.

Also incorporated into the model are defense measures against remotely introduced rogue programs and a cost-benefit analysis that determined that such defenses for a specific environment were cost-justified.

TABLE OF CONTENTS

LISTING OF FIGURES

Chapter 1 INTRODUCTION

Rogue programs[2], including viruses, worms, and trojan horses, have existed for some time[3]. Writers have devoted periodicals[4], security journals , newspapers , and entire books[7] to rogue programs. Rogue programs continue to attack computer systems[8] as well as local area networks (LANs) . Rogue programs will continue to thrive as long as operating systems[1] vulnerabilities exist and LANs are proliferating. Currently, there are over 4000 rogue programs[1] and 93.2% of all installed PCs are expected to be networked. Also, wireless LANs, which were first introduced in 1985[1] , show promise. The wireless LAN market generated about $3 million in 1990, some $10 million in 1991, and $40 million in 1992. Forecasts for 1997 range from over $200 million to nearly $1 billion[1] .

This dissertation shows that inadequately protected wireless LANs are more vulnerable to rogue programs attack than traditional LANs because wireless LANs have not only the same risks as traditional LANs but also have the risks associated with open transmission mediums (radio waves). People who want to insert rogue programs into wireless LANs can scan radio waves and intercept, analyze, decipher, and reinsert data into

the transmission medium.

An abstract model of the rogue code insertion process demonstrates this claim. The abstract model is general and applies to widely used target environments such as the UNIX, Macintosh and DOS operating systems. The model is instantiated on a DOS-based system that uses radio frequency (RF) and employs a Local Area Wireless Network (LAWN) product. The insertion is received undetected, without errors and later executed surreptitiously by the targeted host.

1.1 Related Work

Although there are numerous articles on wireless LANs, only one by Lathrop discusses their vulnerabilities[1]. Lathrop's paper provides an overview of wireless LANs and concludes that wireless LANs face not only all of the risks associated with traditional cable-based LANs but also the additional risk that an open transmission medium imposes.

This dissertation is the first to develop an abstract model of the rogue code insertion process into a targeted network and then instantiates it on a personal computer system. This abstract model has three components: parameters and

requirements definitions, defensive measures and a cost-benefit analysis.

The required parameters and requiremants definition component of the abstract model is analogous to the method used by the Internet worm to attack hosts. Whereas the Internet worm consisted of two parts, a main program and a bootstrap program, the abstract model uses three modules (prober, activator and trigger modules) for basically the same purpose. See citations[1] for a detailed discussion of the Internet worm.

1.1.1 Intrusion Detection Systems (IDSS)

Defensive measures and the access vulnerability likelihood (VL) of the cost-benefit analysis of the abstract model are similar to intrusion detection systems (IDSS). IDSS monitor access control; the VL is used to perform a quantitative analysis. For example, IDSS monitor user activity continuously to detect any suspicious activity as it occurs[2] by comparing a user's current behavior to his/her historical behavior. The VL is used to compute how the rogue code infiltrates the computer system. Accessibility issues include

topological, vector and functional factors. Both are used to prevent unauthorized access; one prevents breaking in; the other (VL) provides the likelihood of breaking in. They are both computer-based security measures. There are currently nine intrusion detection systems in use[2] .

1. Multics Intrusion Detection and Alerting System (MIDAS)
2. Intrusion Detection Expert System (IDES)
3. ComputerWatch Audit Reduction Tool
4. Haystack
5. Information Security Officer's Assistant (ISOA)
6. Network Anomaly Detection and Intrusion Reporter (NADIR)
7. Network Security Monitor (NSM)
8. W&S
9. Distributed Intrusion Detection System (DIDS)

The **Multics Intrusion Detection and Alerting System (MIDAS)**[2] was developed by the National Computer Security Center to monitor the government Multics system; it has been operational since 1988 and it encodes "a priori" heuristic rules that define an intrusion. Midas' rules attempt to detect all penetrations including rogue program infection and misuse.

MIDAS accomplishes this detection by using four types of heuristic rules:

1. Rule 1 deals with current behavior to detect those actions which in themselves (e.g., in isolation) may appear suspicious.
2. Rule 2 uses statistical user profiles to detect any action which deviates from the user's observed recorded past behavior. These profiles list the operator's commonly used commands, typing speed, normal access times, and location.
3. Rule 3 contains a global system profile, which characterizes the normal use of the system. For example, excessive use of the copy command would indicate suspicious activity.
4. Rule 4 sequences commands which characterize known or postulated rogue program attacks. Hence, such attacks can be detected prior to causing any damage.

Currently, MIDAS monitors the use of Dockmaster. Note that although MIDAS is implemented on the Multics system, with some modifications and changing of rules (depending on the system used), it can supposedly be adapted to any system.

The Intrusion Detection Expert System (IDES)[2], being developed at SRI's Computer Science Laboratory since 1985, uses statistical algorithms to observe user behavior to detect any anomaly from the accepted documented normal profile. IDES adaptively learns what is normal for both individual users and overall system behavior. It also uses an expert system that encodes known intrusion scenarios, known system vulnerabilities, and other violations of a system's designed security policy. IDES discerns suspicious activity via a rule base. IDES has been completely redesigned to accomplish its intended objectives. It is modular, extensible, capable of monitoring both heterogeneous and homogenous target machines, and providing protection in a real-time mode. SRI is in the process of enhancing its current IDES prototype, implemented in 1988 to provide a device which will become a tamper-resistant, fault-tolerant, extensible, parallel, and distributed prototype version. This new version will supposedly be more robust, reliable and powerful than the current version. There are currently three versions of IDES:

1. The basic IDES system, which detects any anomalous system activity based on user profiles;

2. The Sun-IDES, which monitors UNIX and uses the C2 Sun Unix Audit Trail; currently in use at SRI as a research prototype

3. FOIMS-IDES, which monitors database use on an IBM mainframe. The FBI has adopted FOIMS-IDES.

IDES endeavors to detect rogue program penetrations and misuse are based on the premise that any exploitation attempts will involve abnormal use of the system. Hence, SRI has accentuated the statistical user profiles and statistical analysis of user activities based on those profiles.

The ComputerWatch Audit Reduction Tool[2], available since September 1989, was developed by AT&T Bell Laboratories. This tool summarizes audit trails and highlights anomalous behavior via detection rules. It is used on a B1 version of the UNIX system V/MLS operating system and detects attempted break-in, masquerading, many types of mistakes by legitimate users, many types of denial-of-service ventures and rogue program penetrations. It can also detect attacks involving more than one person.

Haystack[2], developed by Los Alamos National Laboratory, is designed to assist system security officers to detect and investigate any type of exploitation via anomalous events, security improprieties, and summarizing the system's audit trails of user behavior. Haystack is considered cost-

effective because it uses Zenith Z-248 and Desktop III PC's. It attempts to detect break-ins, masquerading, any type of operating system penetration, denial-of-service, and various forms of malicious use.

The **Information Security Officer's Assistant (ISOA)**[2] was developed by Planning Research Corporation (PRC). It is a functional real-time application prototype which uses a set of statistical tools, an expert system, and a hierarchical set to perform automated auditing and network monitoring. ISOA compares the incoming audit data with a set of expected events. It attempts to detect break-ins, masquerading, and many types of wrongdoing by legitimate users; PRC is in the process of including denial-of-service and rogue program penetration detection. The ISOA is currently used with UNIX Sun operating system C2, and the IBM AT XENIX.

The **Network Anomaly Detection and Intrusion Reporter (NADIR)**[2], operational since August 1989, was developed by Los Alamos National Laboratory (LANL). It aids security managers to detect computer abuse and penetration and attempts to detect break-ins, denial-of-service, many types of automated attacks, and many kinds of legitimate users' abuses. NADIR is specifically designed for use at LANL.

The **Network Security Monitor (NSM)**[2], developed by University of California, Davis, is a research project to detect many types of misuse of hosts connected by a LAN. The NSM prototype is currently running on a Sun 3/50. The target system is the Ethernet and all the hosts connected to it. NSM will intercept all message traffic, regardless of its destination, for examination. Experimentation on a live LAN is anticipated, as well as broadening NSM application to WANs and other platforms.

W&S[3], developed by LANL, is a computer security anomaly detection system. Its inception dates to November 1984. There are currently two versions of W&S in use at the Department of Energy (DOE) and at the National Computer Security Center (NCSC). A third version is in experimental use at LANL. W&S detects anomalies by identifying usage patterns that differ from historical norms and compares current system activity audit records to rules describing past behavior patterns. W&S is especially effective in detecting rogue program penetrations. It also detects other security breach attempts similar to the methods used by the aforementioned systems.

The **Distributed Intrusion Detection System (DIDS)**[3], developed by Lawrence Livermore National Laboratory with participation by the University of California, Davis, and Haystack Laboratories in Austin, Texas, differs from the other IDSSs in that it examines activity on all directly "monitored" hosts on the network while simultaneously examining network activity itself. It has been in Beta testing since July 1992. DIDS has four major components:

1. The host monitor, which resides on each host computer on the network, continuously monitors user activity by comparing that activity with user profiles or particular "signatures" of intrusive behavior, such as reading or writing files.
2. The network monitor provides similar functions as the host monitor on the network.
3. The DIDS director and its expert system (ES) examines the anomalous behavior or suspicious signatures for legitimacy. The DIDS director notifies the user of any unauthorized intrusions.
4. A user interface displays the network's security state, including the level of suspicious activity inferred by the DIDS director.

1.1.2 Cost-Benefit Analysis

The rationale of cost-benefit analysis is that when considering a proposed technology, the costs and benefits to be expected from its implementation should be assessed; and then the technology or improving it is adopted only if the anticipated benefits outweigh the anticipated costs. The implementation of this analysis will vary in accordance with stated assumptions[3]. There are a number of cost analysis methodologies available[3].

Safeguards cost in the cost-benefit analysis of the abstract model was adapted from formulas that Fred Cohen[3] devised to describe the total costs per year of rogue program defenses. Cohen evaluated the costs of today's widely used defenses such as scanners[3], monitors, cryptographic checksums and integrity shells[3]. His 20 costs elements were reorganized and condensed into 7 elements and 5 sub-elements to satisfy the requirements of this dissertation.

Some recurring costs were incorporated from Linda Rutledge's paper[3] to determine communication costs. She proposed a new method for secure transmission, called the Reference Matrix, which provides a technique for encoding a message over public switched networks using a spatial transformation. The re-

curring costs in her cost comparison of the Reference Matrix and other security methods were used to determine the communication costs in the cost-benefit component of the abstract model. Cost-benefit analysis techniques are based on traditional cost-benefit analysis approaches[4].

1.2 Summary of the Contribution

This dissertation makes three major contributions.

1. Demonstrates the problem: By successfully inserting rogue code into a wireless network, this dissertation demonstrates that inadequately protected wireless LANs are more vulnerable to rogue program insertion than traditional LANs.

2. Models a solution and illustrates the instantiation of the solution: This dissertation presents an abstract model that models the process whereby someone uses RF to insert a rogue code into a targeted host's communication data stream. The abstract model is then instantiated into a DOS-based wireless communications system using RF.

3. Provides cost-benefit analysis: This dissertation analyzes the cost of safeguarding the wireless LAN or leaving it unprotected and concludes that (for specific measures) it is cost-effective to implement controls to protect the LAN.

1.3 Organization of the Dissertation

The remainder of the dissertation is divided into four chapters. Chapter 2 builds on previous research related to rogue program characteristics, computer networks and their vulnerabilities, wireless LANs, and the trends that increase the feasibility of remotely inserting rogue code via RF.

Chapter 3 develops an abstract model that shows how the rogue code is inserted into a targeted host using a RF communication channel. The chapter discusses the reasons why the abstract model is developed, how to process and verify the model's instantiated attack mechanisms, and the components that comprise the model: the parameters and requirements necessary to apply it to widely used environments; defensive measures and the cost-benefit analysis that determines when such measures are cost effective.

Chapter 4 instantiates the model on a DOS-based system using a Local Area Wireless Network (LAWN) connection to insert a rogue program via RF into a targeted host on a wireless LAN.

The last chapter concludes that insufficiently safeguarded wireless LANs are more vulnerable to a rogue program attack than traditional LANs. This chapter also concludes that the abstract model developed in chapter 3 can be instantiated, as chapter 4 demonstrated and suggests the advisability of conducting research to protect related systems such as cellular phone penetration vulnerabilities, automatic teller penetration techniques, short wave vulnerabilities, electronic warfare, and satellite manipulation applications.

Chapter 2 BACKGROUND

Chapter 2 provides background material to help the reader understand chapters three through five. This chapter has five sections: section 2.1 delineates the rogue program characteristics; section 2.2 discusses computer networks and their vulnerabilities; section 2.3 discusses and describes wireless LANs; section 2.4 describes trends that increase the feasibility of inserting rogue code remotely, and section 2.5 summarizes the chapter.

2.1 Rogue Program Capabilities

A rogue program must generally have three essential capabilities to infect programs or entire systems effectively. First, because infecting a single file may be inconsequential to some users, the rogue program may be able to replicate itself to multiple files. Second, the program must execute its code to spread the infection. This contamination may be accomplished by either executing an infected program or executing the rogue program code via the operating system's resources such as booting up. Third, the rogue program code may carry a payload to effect whatever task for which the rogue program code was designed. In many cases, rogue programs modify a bonafide program to satisfy the above

capabilities. Sections 2.1.1 to 2.1.4 describe how rogue programs infiltrate hosts, how they work, what they look like, and how they attack.

2.1.1 Rogue Program Infiltration

There are many ways in which an intruder can infect a standalone computer or a network node with a rogue program. Anytime a program is not written by the user himself (or is written by the user but has bugs), and is executed, there is the possibility of it being malicious. When a user gives another user or another machine access to his system, he* is risking infection. Computer systems are infected via an infected disk which is physically placed into the system, or via a remote transfer mechanism, such as electronic mail. The initial infection of a system can occur by:

1. booting a machine with an infected disk
2. copying and/or executing infected software, which may be loaded from diskettes, obtained over a network connection, or via modem on other input methods, such as tape.

---

* "he" generically denotes a male or a female user

2.1.2 How Rogue Prog. ams Work

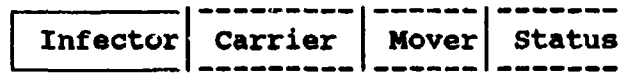
For this section and the rest of this dissertation, unless stated otherwise, the IBM platform is the computing system. When an infected program is loaded and executed in the main memory of the computer system, it can infect other executable programs such as COM, EXE, SYS and OVL files. While executing, the rogue program surreptitiously directs the operating system to append or insert a copy of the rogue code into other programs. Then, when the newly infected program is itself loaded and executed, the rogue code takes control and performs its preprogrammed functions, which generally include self-propagation as well as performing mischievous or destructive manipulations. Depending on the specific type of rogua program, the rogue code may:

1. Remain in main memory
2. Hide in secondary memory such as a hard or floppy disk, etc. Likely hiding spots include executables, the boot sector, root directory, bad sectors, and the partition table.

2.1.3 The Rogue Program's Structure

The one common characteristic of all rogue programs is that they modify or insert an entity such as a program, data, or operating system into a targeted host. This section has a

rogue program's modular structure as modified from citation[4] which is valid for all rogue programs, except for worms:



The above design and ordering of components are used for convenience. In practice, a rogue program may neither be modularly structured nor arranged in any specific order. What is important are the following functional components.

1. The infector component is the rogue program kernel which contains the rogue code. This component contains all the routines and functions to target and attack potential victims, to trigger how much damage to inflict, to identify propagation avenues, and to evade capture/detection.

2. The carrier component is optional; it is simply a normal program within which the rogue program code has been planted. It is useful, however, because it provides the rogue program with a vehicle to propagate to other programs.

3. The mover component is also optional. It moves data which the rogue program has replaced so that the program may still execute normally. The mover component is used with nonoverwriting rogue programs, which will be discussed in section 2.1.4.

4. The status component is also optional and contains a status flag that prevents multiple reinfections. The status flag, which can be a single bit, indicates whether the program has already been infected and stops multiple infections of a single file. The status component will not reinfect an already infected file because reinfections increase the file size making the rogue program susceptible to detection.

For the above structure to be viable, the rogue program must have read and write privileges as well as a means to determine which programs are present. The operating system already contains the required mechanisms for the rogue program to accomplish its purpose. For example, all operating systems provide basic functions such as the COPY, ERASE, TYPE, DIR, PRINT, ATTRIBUTE and PROMPT commands to manage files and programs. Moreover, on DOS systems, all users can have access to the Basic Input Output System (BIOS) and DOS services via software interrupts[4].

2.1.4 Rogue Program Attack Mechanisms

Rogue programs are either overwriting or nonoverwriting rogue programs.

Overwriting rogue programs write over the host program's code, destroying all or part of it (**Figure 1**). The host program may not properly execute after infection.

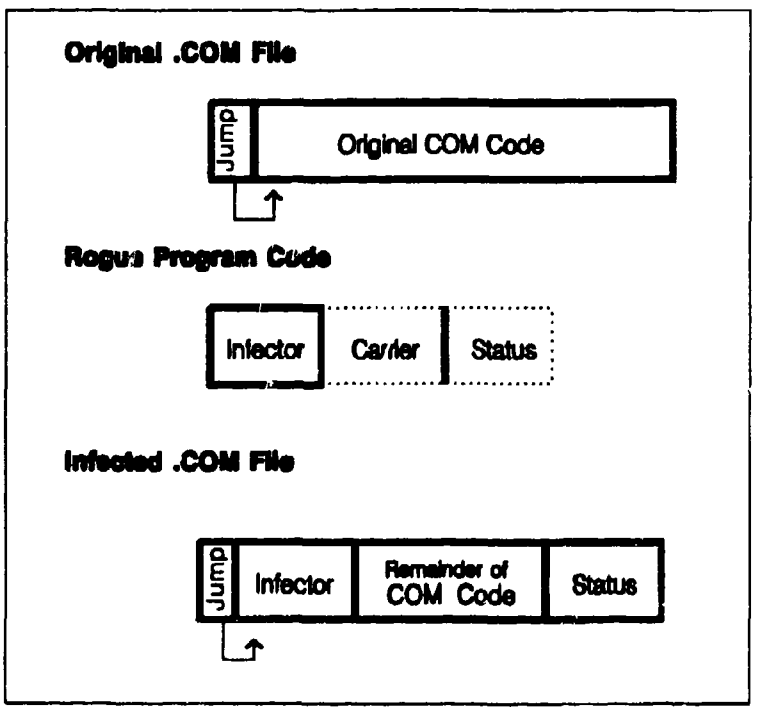


Figure 1. Overwriting .COM File Infector

Nonoverwriting rogue programs substitute the host program's code with their own. In this case, the host program's code can be partially or wholly relocated (Figure 2), and the host program should continue to function properly.

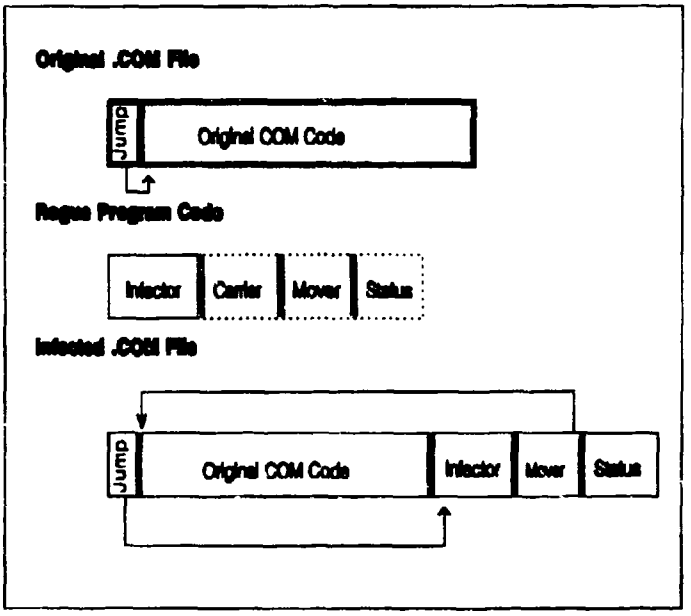


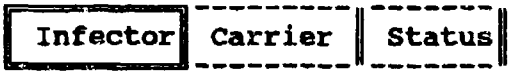
Figure 2. Nonoverwriting .COM File Infector

Although overwriting rogue programs are destructive, they are perhaps the easiest to design and detect[43]. These rogue programs generally overwrite a number of bytes of an executable file so that users cannot recover the file. Since the rogue program overwrites a portion of the host program, the mover component is not required. The following sequence

of diagrams illustrate the operational aspects of such a rogue program[44].

First, assuming that a program (carrier) is already infected, the diagram below displays the infected program and two other programs which are not yet infected:

Rogue Program Code:

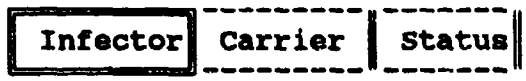


Uninfected User Program 1:

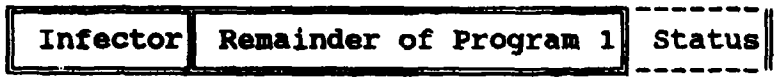
Uninfected User Program 2:

When the infected program is executed, the infector component attempts to infect another program. Once it locates an executable program, in this case, User Program 1, it checks the status component, to determine if it is already infected. If the flag indicates that it is not infected, then User Program 1 is targeted for infection and the rogue code overwrites the initial bytes in User Program 1. The files now appear as:

Rogue Program Code:



Infected User Program 1:

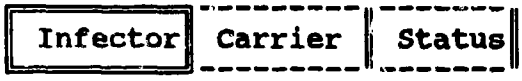


Uninfected User Program 2:

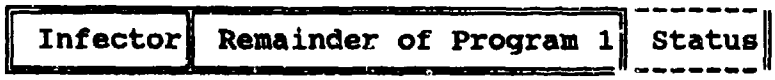
At the conclusion of the infection process, the infector component may trigger a damaging function. Execution then returns to the carrier program so that the program looks normal to users and they will remain unaware of the intrusion.

The infection of User Program 2 follows the same sequence; therefore, its infected structure looks like Infected User Program 1's structure:

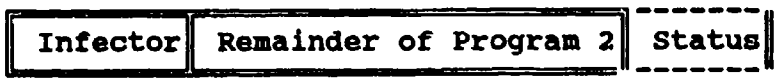
Rogue Program Code:



Infected User Program 1:



Infected User Program 2:



Such an infected program will probably malfunction because the rogue program has overwritten some of its code. The 405 virus, which affects COM files is an example of such a rogue program[45]. The virus overwrites the first 405 bytes of the victim file, and if the victim file is shorter than 405 bytes, the virus increases the file to 405 bytes.

Nonoverwriting rogue programs are the most common[46]. The terminology, however, is deceiving. Although its title implies nondestructive behavior, this type of rogue program can be more destructive than damaging overwriting programs because the overwriting programs generally cause errors immediately with infected executables, and nonoverwriting programs can be present and active within a system for long periods of time without being detected. Nonoverwriting rogue programs have a design similar to their overwriting counterparts; their structure differs only by the mover component, which is the mechanism by which the rogue code is copied to the victim file. This type of rogue program adds code to the host program either by increasing the file's size or by exploiting unused space.

An intruder has many techniques to insert a nonoverwriting rogue code into a host program. Assuming that a program (carrier) is already infected, one such technique operates as follows:

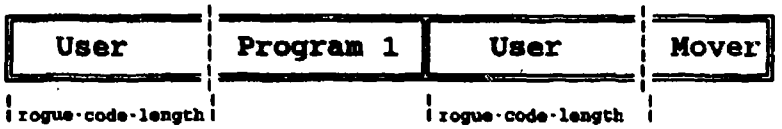
Rogue Program Code:

Uninfected User Program 1:

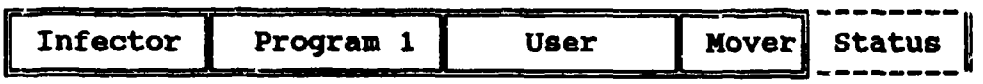
If a carrier program is infected, the infector component attempts to infect User Program 1. It first checks the status component to determine if it is already infected. If the flag is not present, the infector component targets User Program 1 by selecting an area at the very beginning of the program which is the same length as the rogue code as illustrated below:

Here, the rogue-code-length is the sum of the lengths of the infector, mover and status components. The rogue program then

activates the mover component to append that specific rogue-code-length area to the end of the program, thereby preserving the original portion of the User Program 1's code. The mover component then appends itself to the end of the user program.



The initial rogue-code-length bytes in the host program are then overwritten with the rogue code. The rogue program triggers its preprogrammed function, and returns execution to the carrier program. The newly infected program is now itself a carrier program. The rogue program does not manifest any activity until the newly infected program is executed. The status of User Program 1 is now the following:



The original beginning of User Program 1 has been retained, so the host program can still execute properly. Once the infected User Program 1 executes, the routine begins again. The rogue program:

1. seeks an area in the beginning of User Program 2, activates mover to copy that section,

2. appends itself to the space provided,

3. activates the task portion of the infector component to execute the preprogrammed task, and

4. tries to infect the next user program, and remains dormant until the program is executed.

Although User Program 1 is infected, after the rogue program code performs its function, the program continues to function normally, making detection nearly impossible unless the user notices that the file size has increased.

This scenario has many variations. For example, a rogue program can place only part of the rogue code in the beginning of the host program and append the rest to the end. The rogue code can place its code anywhere in the host program, although placing it in areas other than the beginning and end is more difficult.

2.2 Computer Networks

Computer Networks generally use some type of cable for their communication media. Citations[47,48,49] adequately define and discuss networks. They can be configured as LANs, Metropolitan Area Networks (MANs) or Wide Area Networks (WANs)[50,51]. Networks provide resource sharing and interconnection. They must also ensure data integrity, secrecy, and service availability[52]. Networks provide data integrity when they protect data from unauthorized destruction or modification. They provide data secrecy when they protect data from unauthorized disclosure, and they provide service availability when they protect the system deliberate performance degradation. Therefore, to ensure integrity, secrecy and service availability, only the authorized users can access the network and data processing must be protected within the system[53].

2.2.1 Network Access

Unlike many stand-alone systems, networks generally use some form of access control such as identification and authentication. These controls ensure that only authorized users have access to the system or system information. While passwords are the oldest and perhaps the most familiar

personal identifiers, other techniques such as biometrics[54] and smartcards[55] are available.

2.2.2 Network System Processing and Vulnerabilities

Incorporating security into the operating system is one way to protect data processing. Operating systems generally provide several security related functions[56] which are generally located in the operating system kernel*[57] where they monitor and protect all operating system accesses and functions.

Some network operating systems vulnerabilities or functions vulnerable to rogue program attack include[58]

1. I/O processing weaknesses,
2. access policy ambiguity, and
3. readily available commercial-off-the-shelf (COTS) programs are vulnerable because there are many of these and so many people use them.

I/O processing becomes vulnerable when, in the interest of fast data transfer, the operating system bypasses the particular functions protection features.

---

* The operating system kernel performs the operating system low-level functions.

The computer industry has found it difficult to establish a fixed, all encompassing network access policy because of problems with accurately defining the difference between isolating users and allowing them to implement the security kernel. It is important to separate users to protect their data, but, it is just as important to provide them access to data to do their job, such as shared access to libraries, utility programs, and common application data.

People implementing operating systems accommodate COTS packages by using "hooks" to install these packages. Any user can find these hooks and use them as trapdoors* to access and infiltrate the system.

Network operating systems can provide security and controls for all programs that run in its environment, but their size and complexity make it difficult to protect them.

Other functions vulnerable to rogue program attack for networks can include:

1. Accessibility. Networks are easily accessible, since there are so many computers interconnected; there are

---

* A trapdoor is a secret access to a software program for debugging and developing functions.

multiple points of attack. The level of security at any node is dependent entirely on whatever security measures (if any) are in place at that particular node.

2. Resource sharing. Generally, if one computer in a network is infected, other computers are also infected.

3. Routing paths. Users can seldom control the routing of their messages.

4. Unknown nodes. As networks continue to proliferate, security measures at the new nodes will become more and more unreliable.

2.3 Use of Wireless LANs

An alternative technology to "cable-bound" LANs is the **wireless LAN**. Wireless LANs free people from the hardware location restrictions as well as to managing and maintaining miles of wires that connect workstations. Wireless LANs provide hardware mobility and flexibility - essential requirements in our highly mobile society. Managers can configure networks to transmit data via RF transmissions. Currently eight companies[59,60,61,62,63,64,65,66] produce wireless LANs.

Figure 3* lists these companies with their specifications:

Product	Cost/Node	Data Rate	Range (Indoor)	Range (Outdoor)	Frequency	Additional H/W Req.	Encryption Chip	BOOT ROM
WAVELAN	$1300	2M bps	5 Miles	800 ft R	902-928Mhz	Yes	optional	optional
Altair	$3000	10M bps	Unlimited (not wireless)	5000 Sq ft R	18Ghz	Yes	No	No
LAWN	$300	1200 bps	500 ft R	10,000 Sq ft R	902-928Mhz	No	No	No
ProxNet	$500	1200 bps	500 ft R	10,000 Sq ft R	902-928Mhz	Yes	No	No
ARLAN	$1500	2M bps	400 ft R	2000 ft R	902-928Mhz	Yes	No	No
RadioLink	$3450	256K bps	5 Miles	800 ft R	902-928Mhz 2400-2483Mhz	No	No	No
BestLan	$1300	2M bps	5 Miles	800 ft R	902-928Mhz	Yes	optional	optional
IBM	unk	1-10M bps	5 Miles	800 ft R	902-928Mhz	Yes	No	No

Figure 3. Wireless LANs and Specifications

* 1. NCR Wireless LAN System (WAVELAN); 2. Motorola's Wireless LAN Network (Altair); 3. O'Neill's Communications' Local Area Wireless Network (LAWN); 4. Proxim Inc.'s ProxNet (alias RangeLAN); 5. Telesystems SLW Inc.'s ARLAN 600 Wireless Network System; 6. California Microwave Inc.'s RadioLink Network; 7. Black Box Corporation's BestLAN; 8. IBM's Wireless LAN (TBA).

Other companies[67], such as BICC Communications of Auburn, Massachusetts, and Photonics Systems Inc. of Northwood, Ohio, manufacture wireless LANs that use infrared rather than RF transmission techniques. Infrared LANs use basically the same technology as remote-control units for television, VCR or stereo which employ the light spectrum to transmit data between nodes. Infrared transmission uses a much higher data rate than its RF counterpart, and unlike RF wireless LANs, is immune to radio interference.

Unlike RF systems, an infrared system requires that its units be in direct line of sight with each other; signals cannot be transmitted through physical barriers, such as walls and furniture.

Wireless LANs, unlike conventional LANs, are more vulnerable to rogue program infection because with enough resources and time, intruders can scan radio waves and intercept, analyze and possibly decode and retransmit data into the communication medium. Although some hosts use wireless LAN modules that use spread spectrum* technology which makes it difficult to intercept data between hosts, all an intruder needs to do is

---

* Spread spectrum radio transmissions distribute the transmitted data across multiple frequencies.

to use one of eight available modules so he does not have to break any code; the correct module does it for him.

Computer, operating system, network standardization, increased use of distributed systems, and computer connectivity enhance the viability of attacking wireless LANs. All an intruder needs is a complete description of the transmission frequencies, modulation, synchronization and coding function (as discussed in the next chapter).

Using RF to communicate among computers is not new. For example, the world's first computer system to utilize ground based radio packet broadcasting for its communication facility was the ALOHANET system at the University of Hawaii in 1970. Another example of using RF technology to communicate is the Captain Midnight Escapades of 1986. The following subsections discuss the these similar technologies.

2.3.1 The ALOHANET

The ALOHANET (**Figure 4**[66]) used packet broadcasting via radio to give seven campuses on four islands access to a central computer in Oahu. Each campus communicated with the central computer by using an FM radio transceiver whose power was

boosted with powerful repeaters. Two distinct 100 Khz channels were employed: an inbound random access channel, since the probability of contention was high, and an outbound broadcasting channel, since contention was minimal. There were no direct station to station communications[69]. The ALOHANET became defunct in 1979 when a wire-based LAN cable was installed.

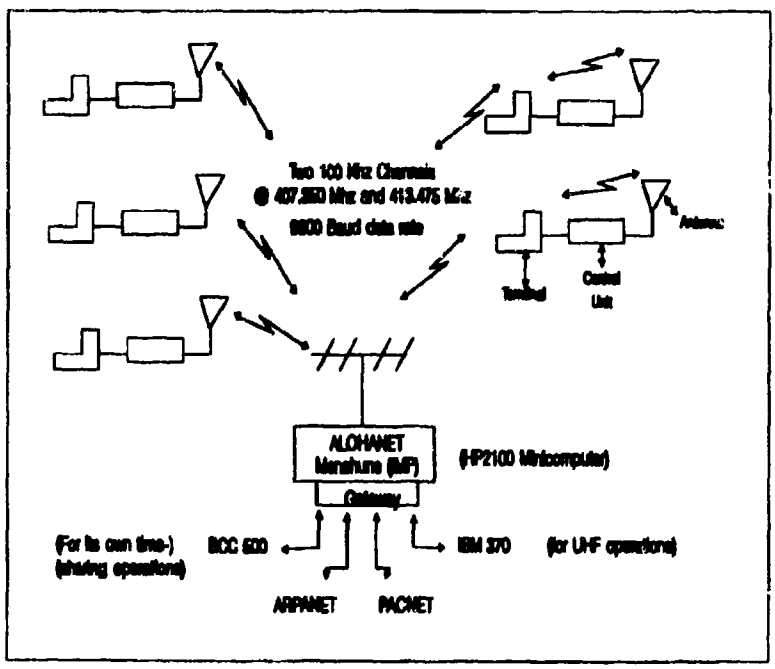


Figure 4. Schematic of the ALOHANET

2.3.2 Captain Midnight Escapades

The Captain Midnight Escapades also used RF technology. Shortly after midnight in April and June 1986, a disgruntled satellite dish user and non-Home Box Office (HBO) cable subscriber preempted HBO with the following message decrying scrambling:

> "Good Evening HBO from Captain Midnight.
> $12.95? No Way! Showtime/The Movie Channel
> Beware!"

Captain Midnight used a transponder at the Central Florida Teleport Co., where he worked, in Ocala, Florida. The transponder consisted of a 10 meter satellite dish with 2000 watts of RF power. It was sufficient to overpower the HBO signal, much to the dismay of the cable company, but to the delight of many satellite dish owners throughout the country[70].

2.4 Trends That Increase The Feasibility of Inserting Rogue Code Remotely

The evolution, development and proliferation of computing networks have significantly enhanced system vulnerabilities to rogue program attacks. Operating system and computer standardization, expanded use of distributed systems, network

connectivity, and wireless LANs have all increased the viability of successful rogue program intrusion.

2.4.1 Operating Systems and Computer Standardisation

Operating systems, because they are standardized, interoperable, transportable, and form a common platform have become a major target for rogue program attacks. Operating systems vulnerabilities are well known. Journals[71], books[72], and periodicals[73,74] delineate ways to identify such vulnerabilities systematically. Figure 5 enumerates some operating system components vulnerable to rogue program attacks.

	Operating Systems		
	DOS	UNIX	Macintosh
Vulnerable to Attack			
File Structure	Files/ Directories	Same	Same (Finder)
System Functions	DOS Functions	Kernel	Resources (Code,CDEV, Patch)
Boot-up Sequence	Boot-up Seq	Same	Init Resource
Command Interpreter	Command.Com	Shell	System File Toolbox
Hidden Files	I/O.SYS MSDOS.SYS	NA	Desktop File
Telecommunication	NA	System Network Utilities (i.e., mail forwarding, authorized access, trusted host files)	NA

Figure 5. Susceptibility of Operating Systems

In addition, books such as Mark Ludwig's The Little Black Book of Computer Viruses[75] teaches the basics of writing rogue programs, complete with examples.

To promote interoperability and transportability and to control acquisition and support costs, standard commercial off-the-shelf (COTS) hardware and software systems that meet national or international standards are becoming more popular[76] than customized hardware and software. Using COTS saves money and reduces the logistical support needed to maintain software. Because the ratio of CPU performance to price doubles every year, it is not cost-effective to develop hardware or software from scratch, which can require up to ten years to develop and deploy[77].

Users must be able to deploy systems rapidly and have access to portable software to adapt quickly to standardization hardware. Hardware standardization includes fixed and floppy drives, controllers, chips, boards, power sources, video cards and monitors, CPU, and many peripherals. Software standardization packages include WordPerfect, Dbase, Lotus 123, Harvard Graphics, plus other management and decision-oriented software packages.

Al[illegible] the features which make COTS hardware and software so appealing are the same features that make these systems vulnerable to rogue code attack. COTS products include a rich spa[illegible] set of functions such as computer architecture

benchmarks, routines, and protocols to provide the maximum flexibility and functionality. But because they are so flexible, intruders, if they can detect a flaw in any of these functions, can access all network nodes.

2.4.2 Distributed Systems Standardization

The computing community including government and corporate services, banking institutions, airlines, and military services, nationwide department stores, and computer stores widely employ distributed systems. Corporations are increasing their use of distributed processing because standards organizations such as the Open Software Foundation (OSF), the International Standards Organization (ISO), and Consultative Committee on International Telegraph and Telephone (CCITT) are endorsing it.

OSF, a consortium founded by Hewlett-Packard, IBM, and Digital Equipment Corp, announced the components of a Distributed Computing Environment (DCE) in May 1990. This environment allows users to run distributed processing applications across a network to allocate the processing power of the networkdynamically. The consortium released the preliminary version of the OSF-Distributed-Computing-Environment technolo-

gy in 1991[78].

The ISO, including constituents of the national standards organizations in the member countries, deals with international standardization of various protocols. CCITT, consisting of national, public and private telecommunication administrations, is primarily concerned with telephone and data communications systems. The ISO and CCITT are both standardizing a framework for structuring distributed applications. ISO is expected to release the Open Distributed Processing (ODP) Draft International Standard which addresses distributed applications in 1995[79].

The "Big Three" software companies, Lotus Development, Microsoft, and WordPerfect, are competing for developers and customers to use their own distributed architectures. Each architectural design is centered around computer systems and interconnectivity, application integration, advanced functionality and common user interface[80].

Stand-alone and distributed systems share some of the same risks; however, distributed system are far more vulnerable than stand-alone systems for the following reasons:

1. Intruders can propagate infection easily because distributed systems are interconnected. For example, one infected machine can contaminate all the machines throughout the communications subsystem.

2. Multiple access points to the connected system and multiple security mechanism levels for each host make installing rogue code easy. The more hosts that there are, the greater their availability, and the greater the likelihood of getting hit with a rogue code, especially if one host is already infected.

3. The availability of a multitude of other services such as network system utilities which include file transfer, remote job entry, and sharing of computing functions provide a rich environment for a rogue program attack.

4. Because distributed systems are harder than stand-alone systems to debug, they require more debugging tools. Sometimes, debugging routines bypass security checks and thereby enhance the system's susceptibility to rogue program attacks.

5. Normally, only a client host distributes new or improved software. When users log-on, as in the case of "prodigy" updates, the host automatically downloads files to the user's machine[81]. Because only one host is

involved, intruders can readily determine system weaknesses.

2.4.3 Enhanced Computer Connectivity

Increased computer connectivity is inherent in standard operating systems, networks, and distributed systems. There were over 400,000 LANs and LAN-operating systems sold in the United States in 1992[82]. Customers probably purchased these LANs to send and receive electronic mail; however, file sharing is expected to play a progressively more important influence, especially with client/server networking*. Therefore, because of connectivity, the whole world may be able to access your electronic door.

2.5 Summary

This chapter described rogue program characteristics, discussed computer networks' susceptibility to rogue code attack, and the use of wireless LANs, trends that increase the feasibility of remotely inserting rogue code by RF and how rogue programs infect wireless LANs. Chapter 3 uses this

---

* The computing system that used to run on a single machine is now a distributed system spread across multiple computers, technologies, geographies, and organizational functions.

knowledge of rogue programs to develop an abstract model of the rogue code insertion process into a communication data stream to a targeted host via radio frequency. Chapter 4 instantiates the abstract model developed in chapter 3 on a DOS-based system.

Chapter 3 REMOTELY INSERTING ROGUE CODE INTO A WIRELESS LAN USING RADIO FREQUENCY

3.1 Introduction

In Chapter 1, the purpose of the dissertation was discussed. In Chapter 2, characteristics of rogue programs were delineated (what they were, how they were structured and how they functioned), networks discussed, and the concept of wireless LANs introduced. This chapter develops an abstract model of the rogue code insertion process into a targeted (networked) wireless communications system using Radio Frequency (RF) atmospheric signal transmission.

The first three sections provide background information and the reasons for developing an abstract model. Section 4 discusses the abstract model. The model is general enough to apply to widely used target environments such as UNIX, Macintosh and DOS operating systems. In Chapter 4, a DOS-based model is used to demonstrate the feasibility of actually inserting rogue code to a targeted host via RF.

This chapter has five sections: an introduction (3.1); background (3.2); reason for the model and its attack mechanisms (3.3); model development (3.4), and summary (3.5).

3.2 Background

Rogue codes can rapidly spread throughout target computer networks[63] which are vulnerable to rogue code attacks. The magnitude of the damage depends on the intruder's intent and the system's safeguards against infection. In October 1989, Cramer and Pratt's "Computer Virus Countermeasures - A New Type of Electronic Warfare," discussed for the first time applying computer rogue program techniques to electronic warfare[64,65]. This dissertation is the first to develop a generic model to model the insertion of rogue code into a targeted system and instantiate it on a DOS-based system.

3.3 Attack Goals

We have so far discussed similar technologies to use RF to effect computer communications. The following subsections discuss the motivation to develop a general model of the rogue code insertion process, rogue code insertion procedures and verifying insertion success.

3.3.1 Motivation to Develop an Abstract Model

The reason to develop an abstract model is to show how easy it can be to insert rogue code into a targeted host via RF. The

purpose of inserting rogue code into a targeted host can include a variety of covert goals including disrupting, degrading or exploiting the targeted host's operational capabilities to function properly. The main components of the model at a conceptual level shown in **Figure 6** are the intruder, the attack mechanism and the targeted host.

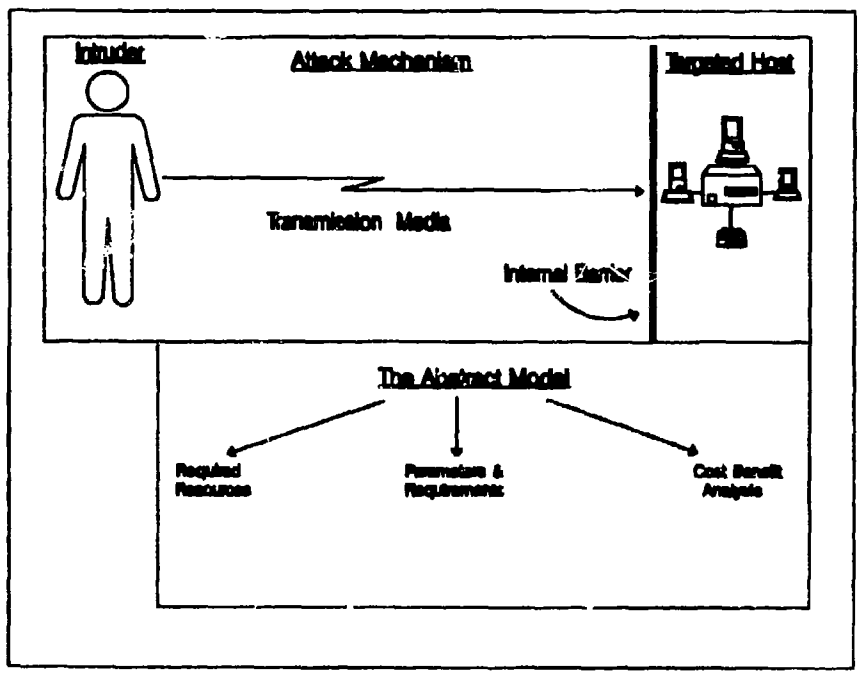


Figure 6. Conceptual Model

To affect a targeted host's operations adversely, the intruder's program must insert, modify or delete the host's control messages. Such manipulations can among other things cause the routing algorithm to select suboptimal routing.

Distributed system nodes can be highly vulnerable to false system control messages, either from communications errors or from a deliberate attack. For example, the Internet is vulnerable to false control messages being inserted either on interswitch communication lines or from one of the switches themselves [switches are the network's connection mechanisms][86]. Flooding the network with a continuous stream of bogus messages can significantly increase processing time and hence disrupt or degrade the system's operations. Packet communications are very vulnerable to a variety of fraudulent message and message alteration attacks because packets can be generated that appear to have come from another source. Packets can be captured, modified and reinserted into a network without the bonafide hosts knowing it.

Other typical approaches to disrupting or degrading a targeted host's operations include forcing a system crash, destroying data, or inserting delays in real time systems. An intruder can easily cause a system crash by modifying a program which executes automatically during the booting procedure such as COMMAND.COM for DOS-based machines. Destroying data is another way of disrupting or degrading the system. An intruder can destroy data by overwriting or erasing the data or by changing the pointers to that data. For example, the

intruder could change the pointers on a DOS-based machine by modifying the FAT. He can alter data either en masse or piecemeal, depending on his goals. He can also disrupt or degrade data by writing a rogue program that delays packets. Disruption and degradation attacks can be referred to as denial of service attacks because they are intended to reduce the communication channel's information carrying capacity.

Intruders can use passive measures in addition to active measures to gain valuable information from a targeted host such as revealing the host's system resources, such as configuration and data and system files, and addresses and listings of trusted hosts to which they are connected. The intruder can exploit this information at the time of the attack or at a later date.

3.3.2 Attack Method

The following "Attack Process Events Time Line" provides the guidelines (**Figure** 7) for examining the attack process:

"Attack Process Events Time Line"

1. Determine Possible Target Hosts
2. Probe Target Characteristics
3. Build A Rogue Program
4. Task the Rogue Program
5. Rogue Program Executes Task at Predetermined Time.

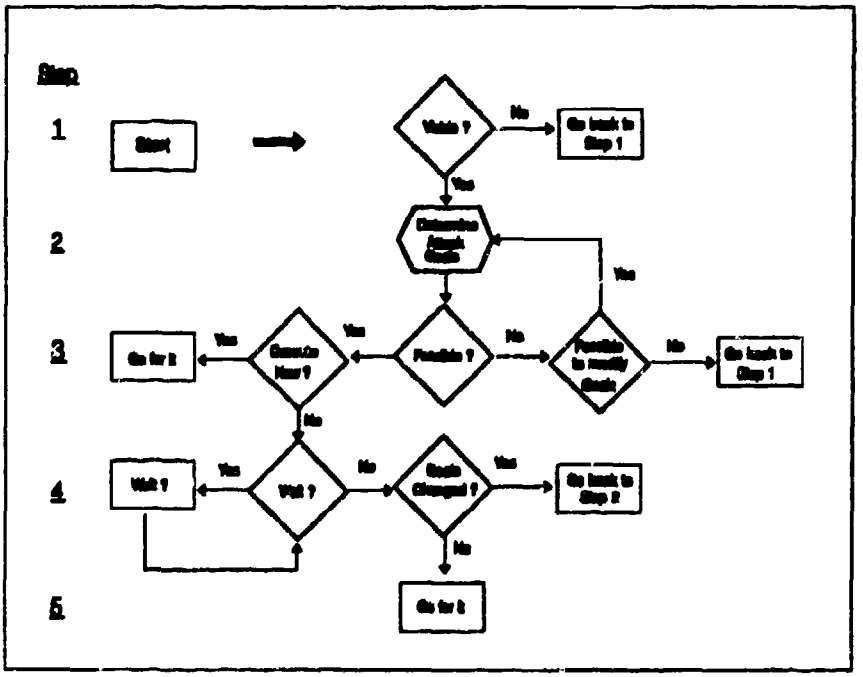


Figure 7. Attack Time Line

First, the intruder must list targeted hosts. Then, he must probe system characteristics and resources to determine if the attack against a specific host is viable. If attacking a specific host is not viable, he will continue to attack other hosts until he is successful. Once he is successful, he must determine his goals. If his goals are feasible, the intruder will build the rogue program code and infect the system with it. If the goals are not feasible, he can modify them. Depending on the intruder's goal, the rogue program can lie dormant for future execution or be executed immediately.

3.3.3 Verifying the Success of Rogue Code Execution

Intruders must give serious consideration to determining if the rogue code is in control. For example, an intruder might build a signal into the program that would respond automatically or upon request to verify that the injected code is operational. The rogue program may send such a signal, which would require a very small bandwidth via covert channels so that the host system could not detect the signal and the infection. However, such a strategy would promote opportunities for detection by the targeted host. Depending on the situation, it may be worth the risk of exposure to obtain confirmation.

Another option available to the attacker to ensure that the rogue program is actually doing what it was designed to do is to conduct covert or overt testing upon infection or periodically after infection.

Covert or overt testing can be conducted with a system similar to the targeted hosts' system in a controlled environment established by the attacker. The authenticity of such a system would depend on how much information was available on the targeted system this "analogous parallel" system may consist of an abbreviated or an exact version of the targeted host. Although intruders can verify that they have infected

the host with the rogue code they can not predict how the rogue code will affect the host, because they do not have full knowledge of software, special hardware, or firmware. In short, there does not appear to be an adequate feedback mechanism to determine the operational status of such rogue programs or to control them once they are executed.

3.4 An Abstract Model - Overview

The abstract modal has three components: required parameters, defense measures, and the a cost benefit analysis. Figure 8 pictorializes the abstract model.

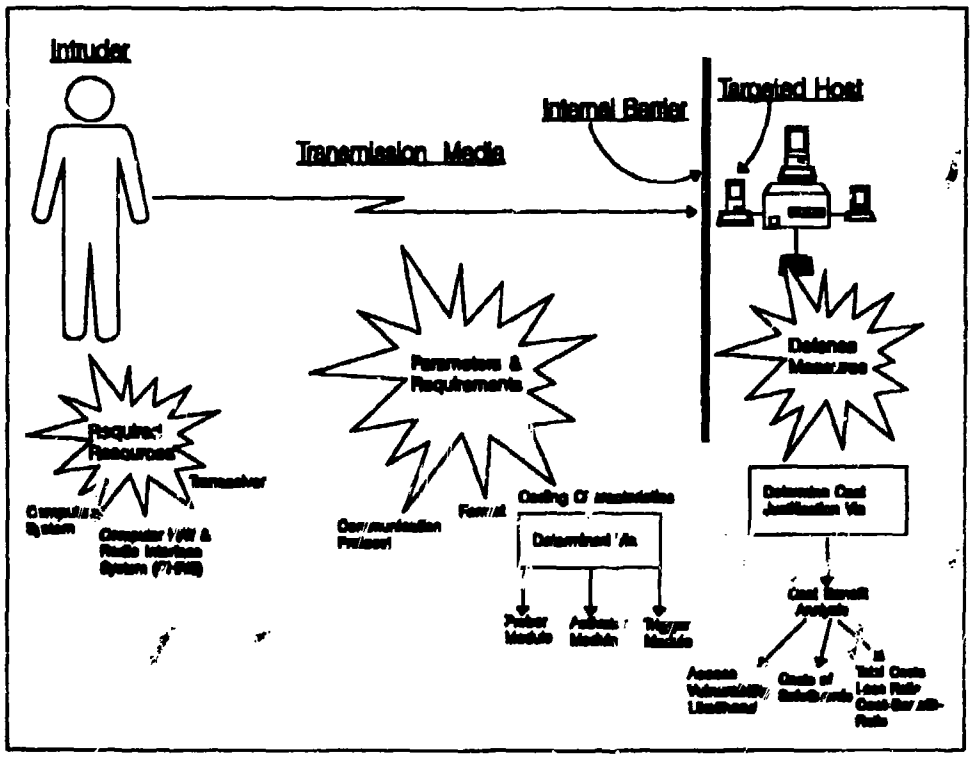


Figure 7. The Abstract Model

3.4.1 Parameters and Requirements

To develop an abstract model it is first necessary to identify the parameters and requirements that exist within a system. Within the context of a network, the key parameters to be considered are the targeted network's communications channel in order to format the rogue code properly so that the

receiving host would accept it and how the target host processed the data it received from a communications link including the target host's protocols and applications.

3.4.1.1 Communications Channel

A network may consist of a series of homogeneous or heterogeneous computers connected in a local area network (LAN) or in a wide area network (WAN). The communication over connecting transmission media is accomplished using complex protocols. Protocols are designed as a series of dependent layers to attenuate their complexity. While the topology of these layers may differ for different networks, the characteristics are basically the same.

The International Organization for Standardization (ISO) has proposed the **Reference Model of Open Systems Interconnection (OSI)**, as a first step toward internationally standardizing the various protocols. The **ISO OSI** reference model, commonly known as **OSI**[87], has seven layers, each one built on the previous layer; each with its own specific function (see **Figure 9**).

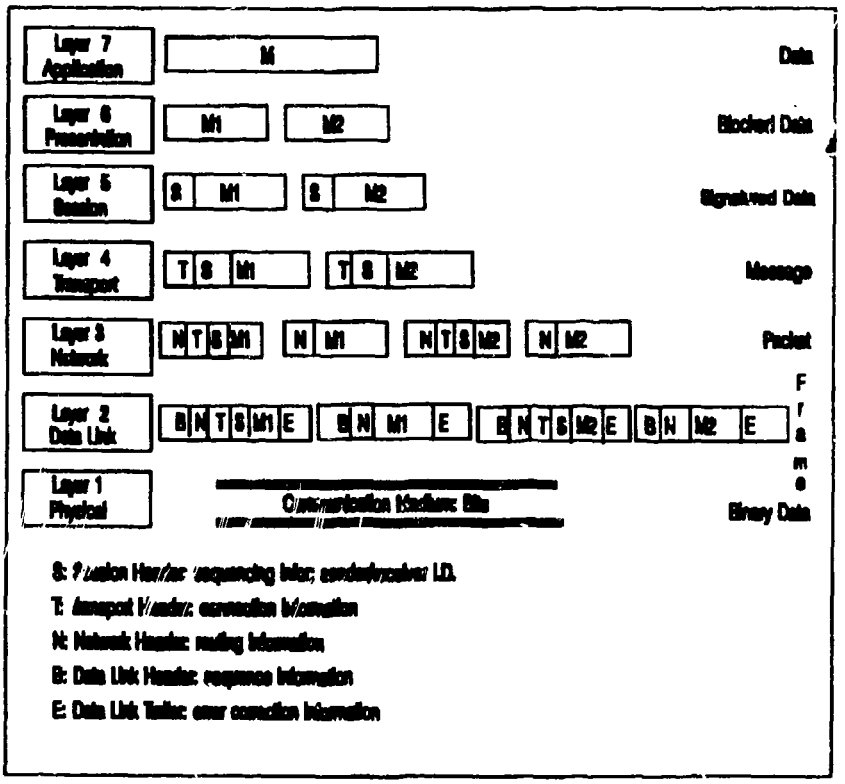


Figure 9. OSI Message Transmission Format

A sender initiates message transmission at layer 7 with an application program. Message transmission traverses the interdependent layers down to layer 1, the physical layer, which is concerned with transmitting specifically formatted, individual bits over a communication channel to the physical layer at the receiver.

While the above series of layers simplify the protocol, they also introduce more opportunity for rogue code programmers to penetrate systems especially since networks were not designed with security as a high priority. See **Figure 10** for OSI vulnerabilities.

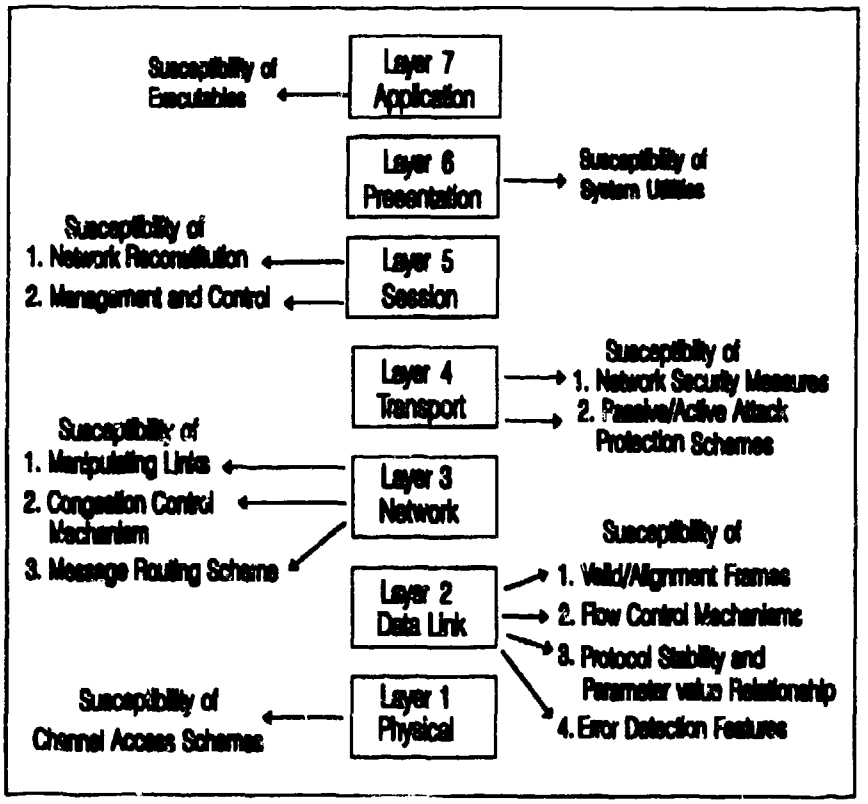
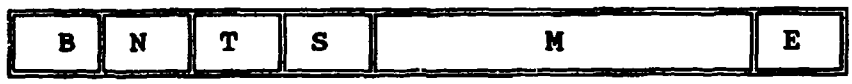


Figure 10. OSI Layer Vulnerabilities

3.4.1.2 Data Stream Conformation

The injected rogue code must be formatted properly for the targeted host to interpret it as normal network data. The ISO OSI model follows this message transmission format:



Where:
- **B** is the Data Link Header - Frame Marker
- **N** is the Network Header - Routing
- **T** is the Transport Header - Priority
- **S** is the Session Header - Synchronization
- **M** is the Message
- **E** is the Data Link Trailer - Error Correction

The intruder must also know the following parameters to be able to insert the rogue code into the data stream effectively:

1. transmission frequency
2. synchronization
3. coding characteristics.

We describe these now.

3.4.1.2.1 Transmission Frequency

Transmission frequencies vary from 300 bits per second to 2 Mbps. The intruder must know this frequency, which determines the message transmission rate, to insert rogue code into a data stream.

3.4.1.2.2 Synchronized Communication

Communication is synchronized when the data characters and bits are transmitted and the sending and receiving hosts are synchronized. For example, when one interface message process (IMP) wants to send a frame to another IMP, it sets the frame in a memory buffer and then starts the transmission hardware. Before sending the first character in the buffer, the transmission hardware sends a synchronizing signal defining the start of the frame. After the message is transmitted, another synchronizing signal is sent to define that the process is completed. There is a finite amount of time allotted for specific packets to be transmitted and acknowledged. Synchronization signals also define the time period that a sending host will wait until a packet is resent if no acknowledgement is received. Such signals define delays among hosts as well. Therefore, to inject rogue code into a data network successfully, the intruder has to know the synchronization signals.

3.4.1.2.3 Coding Characteristics

Coding characteristics include transfer modes, such as ASCII and octet, packet size, and other structural parameters. The intruder must know these characteristics to make the rogue code look like the code it is replacing; otherwise the receiving host will reject it.

3.4.1.3 Code Generation

Unlike standard methods of inserting rogue code into target systems where code size is not critical, the rogue code to be inserted via RF should be small. The time period within which such code can be injected during a transmission is limited. Hence, limited insertion time dictates limited code size. A packet size code of 512 bytes or less would be optimal. Note that the smallest known rogue program, the "Define Virus", is only 30 bytes[4].

Intruders can compress the rogue code to make it as small as possible. After they inject this compressed code into the targeted data stream, and the "duped" host accepts it, it can decompress and infect the targeted host while remaining inconspicuous. The following modules help the reader understand how an intruder can compress rogue code and infect computer systems:

Prober Module	Activator Module	Trigger Module

The **Prober Module** ascertains the target host's specific characteristics and gives them to the Activator Module. The

Activator Module contains the "builder," which collects and analyzes all data to build or create the rogue code, to create a rogue program which uses the targeted system's own resources. The **Trigger Module** executes the assigned tasks and propagates at will by using stealth techniques as discussed in chapter 1 (hides, infiltrates, bypasses antirogue program tools in place).

The Prober Module (**Figure 11**) decompresses (if the code is compressed) and initiates its probing function.

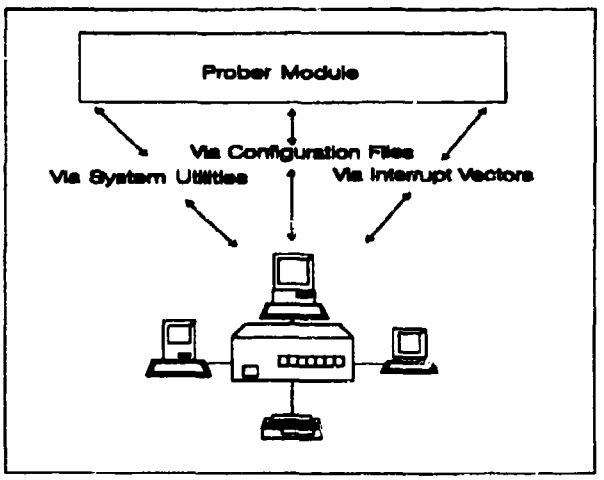


Figure 11. Prober Module

The abstract model's prober is instantiated by the Internet Worm's prober module functions; they both probe the targeted machines in the network for information using system utilities, public configuration files and the target's

interrupt vectors.

If the probing module is successful, it performs the following sequence:

1. makes a copy of the original, compressed code
2. stores the compressed code in high memory for DOS machines or in another file for other platforms
3. "feeds" the information it has obtained to the Activator Module so that the Activator Module can compile the rogue code
4. searches for other viable paths to propagate the rogue code to other hosts.

If the probing module successfully performs this sequence, it transmits the original, compressed code to identified hosts and deletes its original copy from memory or deletes the file within which it used to hide.

If the probing module is unsuccessful, it continues its attempt to perform the sequence with the exception that it does not feed information to the Activator Module. Each time it fails, it deletes itself from the particular network it is trying to invade and keeps trying to invade targeted networks until it is successful.

The Prober Module activates the **Activator Module (Figure 12)** which consists of a decompressor unit, a compiler and groper units. The decompressor unit decompresses the compressed rogue code. The "builder" unit compiles its code to build the rogue program, and the groper unit seeks procedural or technical vulnerabilities, such as poor passwords which will be feed into the Trigger Module.

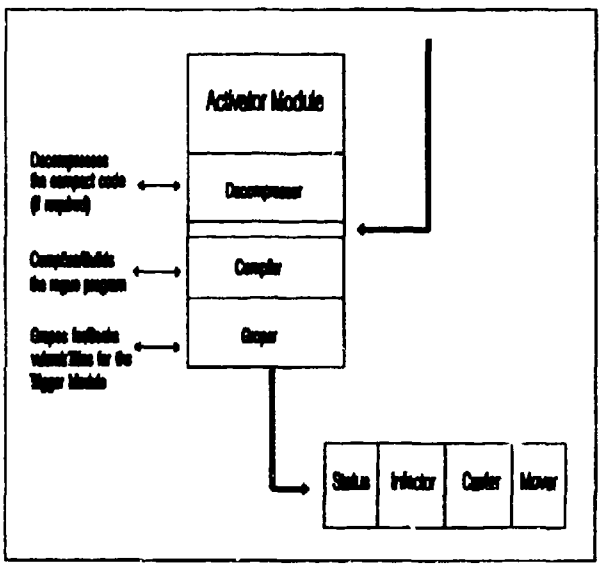


Figure 12. Activator Module

The **Trigger Module (Figure 13)** executes the rogue program to infect the targeted system. It uses stealth techniques to infiltrate the system, bypass any resident antirogue program

tools, camouflage itself, and propagate at will.

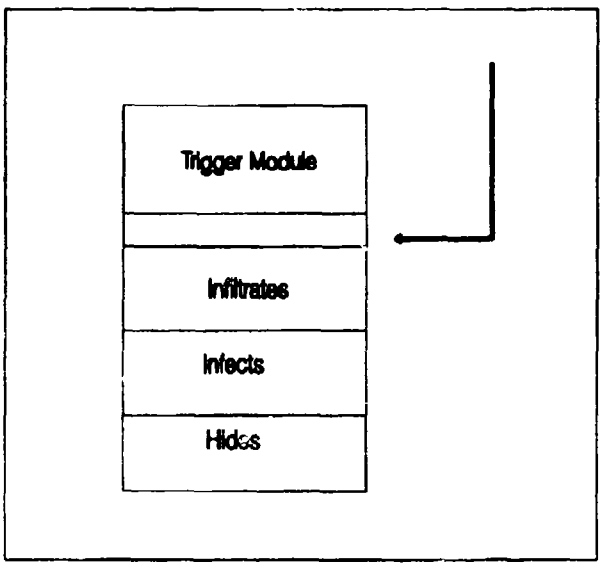


Figure 13. Trigger Module

The technique of injecting rogue code into the target's host data stream may include creating a virus, while subsequent propagations may include transmitting a worm. Viruses do not require network connectivity as worms do, and can therefore access more machines than worms. This dissertation's attack uses a virus to infiltrate the targeted system. **Figure 14** illustrates the Generic Rogue Program Insertion Model and pictorializes direct and indirect rogue code attack mechanisms.

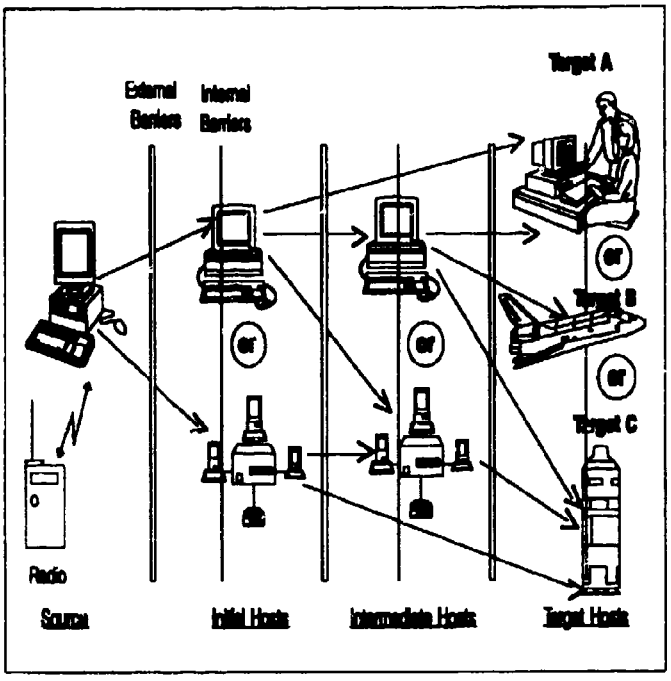


Figure 14. Insertion Module

3.4.1.4 Required Resources

To ascertain the required parameters as discussed above and insert rogue code via RF into a targeted system, the model uses "ham-radio" technology. The following three hardware components are required to insert rogue code into an RF data stream:

1. a computer system
2. a computer hardware/radio interface system (CHRIS)
3. a transceiver.

It is assumed that the message is in clear text. Otherwise the intruder would have to capture and decrypt the message to determine all parameters, and insert the rogue code into the system.

The intruder uses commercially available and inexpensive hardware components to insert rogue via RF into a computer system (**Figure 15**).

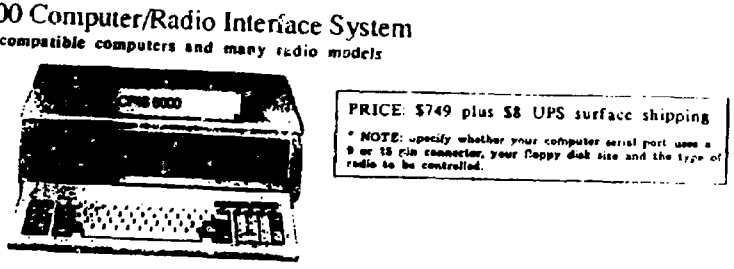
The underlying technology is readily available – only three components required:

1. Computer System – IBM XT/AT or compatible with modem, running DOS 2.11 or higher;

< $500.00

2. Computer Hardware/Radio Interface System (CHRIS) – the interface device between the computer terminal and RF transceiver. The interface assembles and disassembles packets and provides error detection;

< $750.00



3. Transceiver (radio) – to transmit and/or receive data;

$250.00 - $5000.00

Total Cost = $1500 - $6250

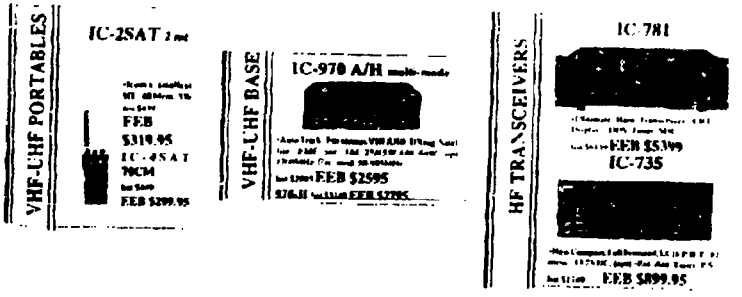


Figure 15. Hardware Resources

The Computer Hardware/Radio Interface System (CHRIS), **Figure 16**, the interface device between the computer terminal and RF transceiver, assembles and disassembles packets and detects errors. The transceiver (radio) transmits and receives data. The intruder uses these three components to intercept and download packets to determine required parameters such as communication protocol, message transmission format, transmission frequency, synchronization, and coding characteristics. The intruder can then replace specified packets with rogue code.

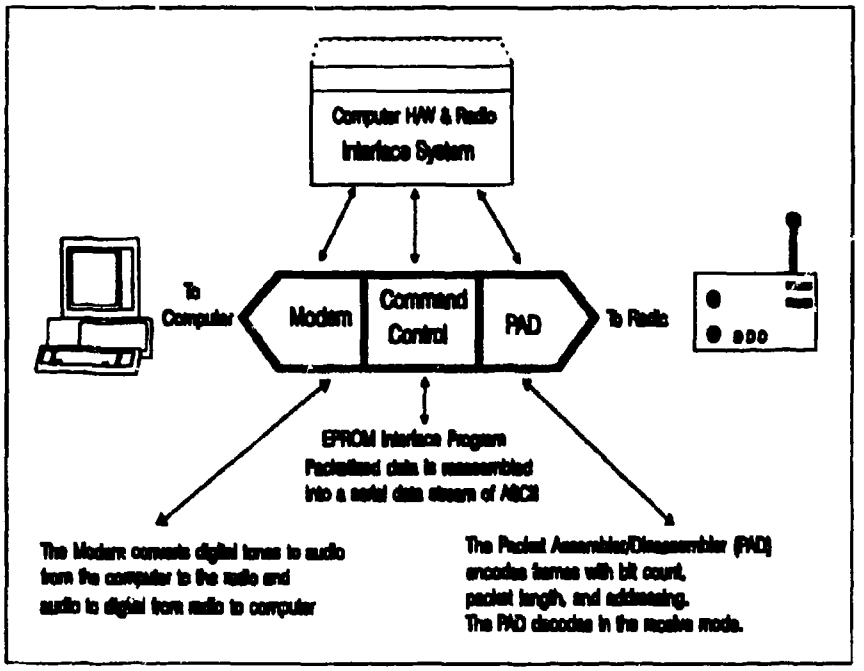


Figure 16. Computer Hardware/Radio Interface System (CHRIS)

RS-232 communications ports make configuring the CHRIS easy. For example, a RS-232 cable allows the CHRIS to interface with the computer and the transceiver. The standard RS-232C serial port consists of up to 25 pins, only a handful of which are required to configure the CHRIS (**Figure 17**).

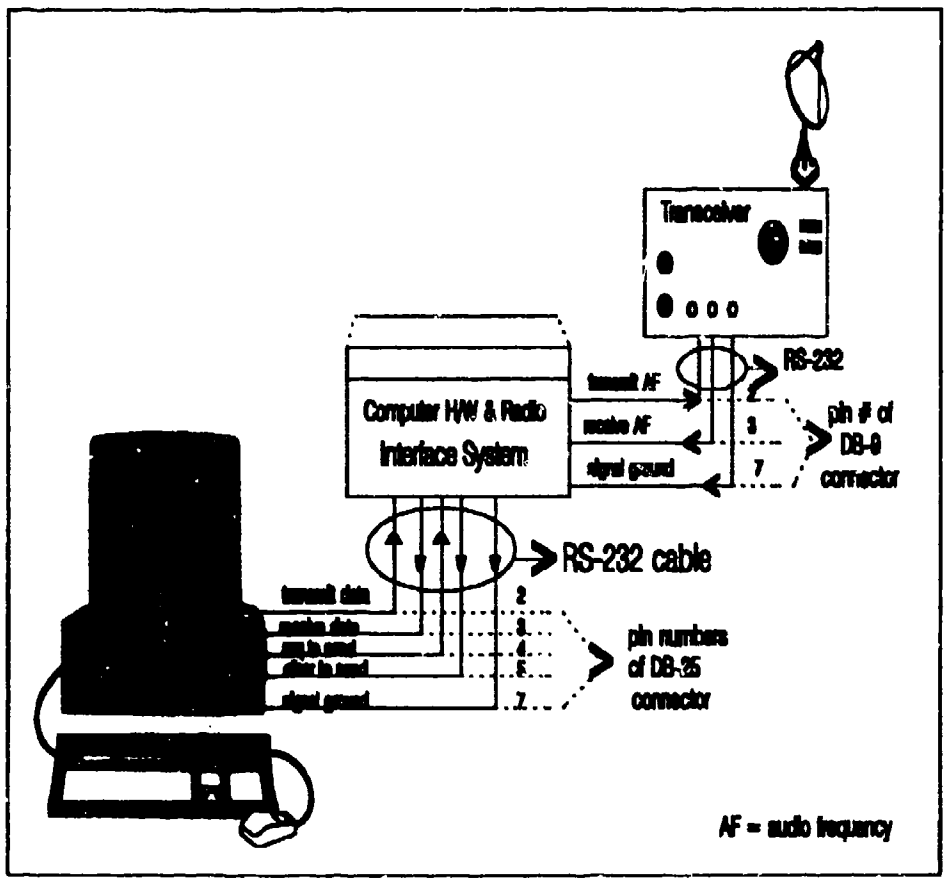


Figure 17. Example Configuration

3.4.2 Defense Measures

While subsection 3.4.1 developed the nucleus of the abstract model, by determining the communication protocol, message transmission format, frequency, synchronization, coding characteristics and required resources, subsections 3.4.2 and 3.4.3 extend the model. In subsection 3.4.2, defensive measures and countermeasures are incorporated into the abstract model to complete it. Further, subsection 3.4.3 addresses whether the defenses are cost justified.

The following six techniques can be used to defend against RF-insertion:

1. cyclic redundancy checks (CRCs)
2. checksums
3. encryption
4. digital signatures
5. built-in security controls
6. combinations of the above

For purposes of illustration, the experiment described in this dissertation uses CRCs and checksums to demonstrate successful detection of rogue code insertion.

The cost-benefit analysis, subsection 3.4.3, indicates if using these protective techniques is cost-justified.

3.4.2.1 CRCs

CRCs check the number of a file's sequential bytes to assign a unique number for that file by treating bits as a representation of a polynomial with coefficients of 0 and 1. For example, a k-bit message is regarded as the coefficient list for a polynomial with one or more k terms, ranging from x^{k-1} to x^0 with a degree of k-1. For example, 101011 represents a six term polynomial with coefficients 1,0,1,0,1,1 or $x^5+x^3+x^1+x^0$. To produce the unique number, polynomial arithmetic is performed using modula 2, in accordance to the rules of algebraic field theory[89]. The three polynomials that are currently the international standards are CRC-12, CRC-16, and CRC-CCITT[90].

CRCs can detect unsophisticated rogue programs, because any change in the number of a file's sequential bytes produces a different CRC. However, sophisticated rogue programs[91] such as those containing stealth capabilities[92] can circumvent CRCs. Therefore, a CRC check alone may not prevent such attacks.

3.4.2.2 Checksums

A checksum calculation is the exact number of a file's individual bytes. The process of performing a checksum verifies a file's integrity prior to execution by making sure

that a file has the exact number of bytes that it should have. Checksum algorithms range from the very simple to ultra-complex. Users can also employ checksums in conjunction with encryption to determine if a file has been modified. A table of checksums for each file can be stored off-line, on a write-protected floppy, in ROM, on a card, or even encrypted somewhere in the system. When a file is loaded, the checksum of the executable file can be compared with the checksum in the table of the file to verify the file's integrity. An assortment of different checksum algorithms exist[93]. Checksum algorithms can detect RF-inserted, non-stealth rogue code.

3.4.2.3 Encryption

Encryption can prevent unauthorized users from gaining access to information. Encryption consists of an algorithm and one or two keys. The algorithm uses a key to scramble the message, called plaintext, into unreadable ciphertext. The same key and the same algorithm unscramble the ciphertext[94]. Encryption, which can both prevent the insertion of rogue code and isolate rogue programs once a system has been infected[95], can also be effective in an RF environment. Regardless of the context within which encryption is used, using encryption mechanisms to transmit messages can make it very difficult for

rogue program writers to insert their rogue code onto a transmission media via RF or any other means because the rogue program writer probably will not have access to the proper key to encipher/decipher the message. Although an encryption algorithm may be breakable[96], it may not be practical to do so, because it would take too long to decrypt it[97,98].

Although encryption can be a powerful tool, it alone may not prevent the insertion of rogue code via RF. Encryption protects against disclosure and detects modification attempts. Using encryption makes a potential rogue writer work harder than if he would have to if code were not encrypted.

3.4.2.4 Digital Signatures

Digital signatures authenticate messages to defend against the threat of rogue code insertion onto a data stream via RF. Digital signatures can be performed at the message or at the packet level in several ways. The three most widely used are the Rivest-Shamir-Adelman (RSA) algorithm, the Data Encryption Standard (DES)-based message authentication code (MAC)[99], and the Digital Signature Standard (DSS)-based Digital Signature Algorithm (DSA)[100].

Digital signatures make it difficult for rogue programs to insert code. Digital signatures at the packet level will make it virtually impossible for rogue program code to be successfully inserted onto a transmission media via RF. The rogue code could be detected - if the imposter does not possess the originator's private key. To limit performance penalties and overhead, digital signatures can be utilized only on the first packet of a message and still ensure reasonable security, nominal performance degradation and lower cost.

3.4.2.5 Safeguards Incorporated in Commercial Wireless LAN Software

The wireless LAN software that comes with many of the commercially available modules incorporates one or more of the following five security mechanisms that make it difficult to insert a covert rogue program and to infect the network nodes in general:

1. The network software can ensure that no two nodes have the same name.
2. The network software can use a security code to authenticate users on the network; the code can be a number from 1 to N, where N can be any number greater than or equal to 1. LAN modules must have the

same security code to talk to each other.

3. A "so-called" secure channel can be set up so that other modules in the network can not intercept messages. Users can choose between two or more channels.

4. Users can purchase an encryption module separately.

5. Users can purchase an optional boot-up from ROM.

6. Many products use spread spectrum technology.

Figure 18, compares the defensive mechanisms available with each of the wireless LAN products discussed in section 2.3. The products are listed in order of the number of defensive controls incorporated.

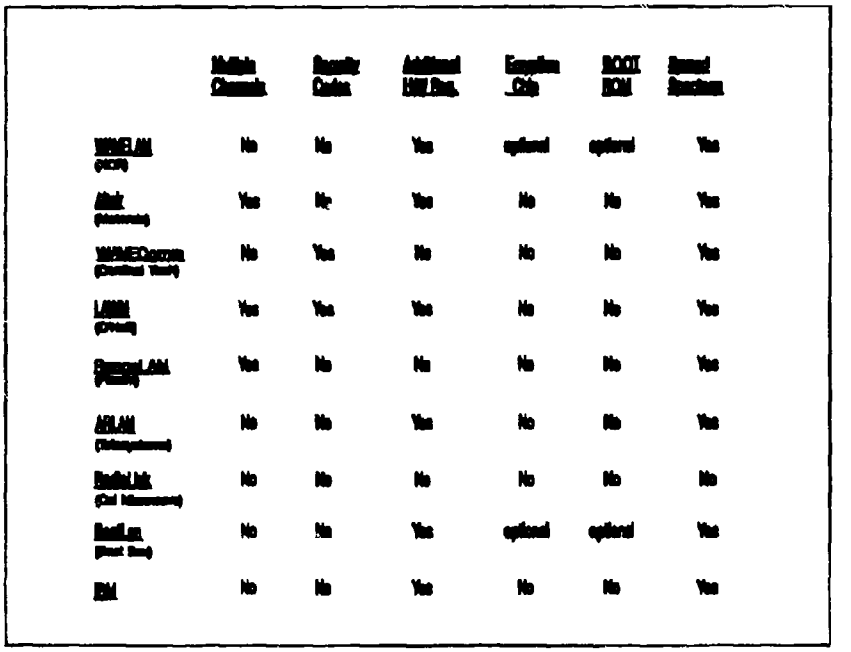


Figure 16. Comparison of Wireless LANs Defensive Mechanisms

These products (except for RadioLink) all use spread spectrum technology and the network software defends against having two nodes with the same name. Spread spectrum technology is incorporated within the wireless LAN modules. If an attacker was using the same module as the bonafide host, spread spectrum is no longer a problem for the attacker. The module does the necessary spreading and decoding of the data. Moreover, although no two hosts on the wireless LAN are

supposed to have the same name, using O'Neill's LAWN package, The author was able to have two hosts with the same name as long as one of those hosts was inactive. In short, an intruder with the same host name as a bonafide user's name could send data to another bonafide user, assuming that all other parameters such as rate of transmission, security code and channel were the same. Note that the only requirement to infecting the system in this way is that the host that the intruder is masquerading as is not transmitting over the wireless LAN. The bonafide user could still be doing other work on the computer and does not have to be logged of.

3.4.2.6 Software and Hardware Mechanisms

Antirogue software products alone may not prevent rogue code from being inserted into a transmission media but may prevent such code from being executed at a targeted host because the

software will have detected the code before it executes. Furthermore, this measure may be effective against only some rogue programs. Stealth rogue programs as defined in citation[101] may be able to bypass some of these control mechanisms, depending on the specific mechanism(s) used.

Antirogue hardware products may be effective against known as well as unknown rogue programs, depending on the product quality and control mechanism(s).

3.4.2.7 Defense Mechanisms Combinations

Finally, a combination of control mechanisms will provide more protection and will make it more difficult for rogue programs to bypass protection schemes. The use of encryption and digital signatures should be considered for incorporation into RF nets; otherwise, it may be possible to compromise each of them alone.

An RF net without any security mechanisms is vulnerable to rogue program attack. All of the above control mechanisms, singularly or combined, implemented either by software, hardware or both, will help protect a communication channel from rogue programs, but at a price. The next section analyzes the cost and benefits of each control mechanisms.

3.4.3 Cost-Benefit Analysis

To determine whether it would cost more to implement a control or to accept the anticipated cost of intrusion, a cost-benefit

analysis can determine whether a specific defensive control's cost is justified. The following cost-benefit analysis is a less comprehensive, less time consuming but more appropriate technique than risk analysis[102,103,104]. The cost-benefit analysis alleviates some of the difficulties in analyzing and evaluating those controls which would reduce the seriousness of rogue infection via RF. It incorporates formulas devised by Fred Cohen[105] that describe the total costs per year of rogue program defenses and by Linda Rutledge[106] that determine communication costs with a proposed access vulnerability likelihood (VL) that we develop. The analysis ascertains a cost-benefit ratio by:

1. determining the accessibility of computing systems to rogue program attacks (VL)
2. determining the yearly cost of applying antirogue products
3. determining the basic cost (BC), recurring cost (RC) and the expected yearly cost of damaging the computing system.

3.4.3.1 Access Vulnerability Likelihood (VL)

The VL is a unit of measure defined as the vulnerability of a closed network resulting from the direct connection to any node in the network and their associated links. A computing system's accessibility to rogue program threats, how the rogue code infiltrates the computer system, includes topological, vector and functional factors (**Figure 19**).

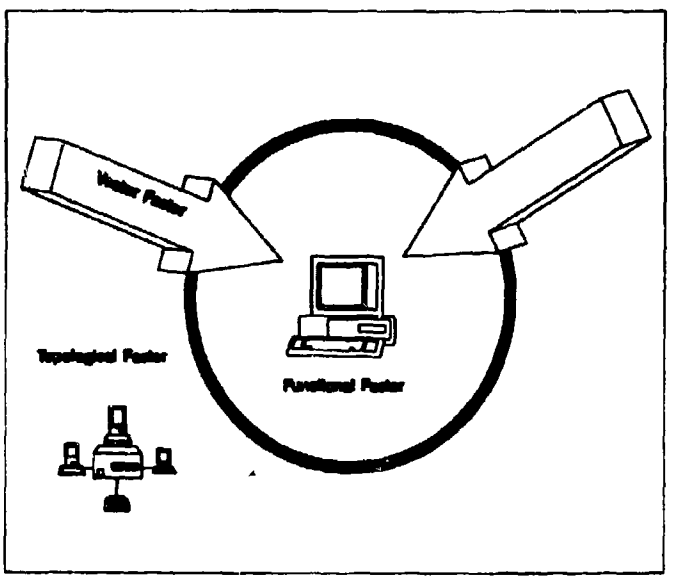


Figure 19. Accessibility Vulnerability Likelihood Components

Topological factors, the physical characteristics of a computing system, connoted by T, consist of connectivity/interface links among computing systems. These

links are potential entry points for infection. Vector factors, connoted by V, consist of carriers that serve as rogue program vectors directly connected from each link. Vector factors are used to determine the likelihood that link interfaces are infected. Functional factors, which may be based on subjective experiences and are assigned weights, connoted by F_i, consist of the likelihood of penetration F_{id} and the havoc* F_{ih} a rogue program can inflict which depends on the presence of or lack of defense mechanisms a computing system incorporates. The Functional Factors, F_i, can be denoted as $F_i = F_{id} * F_{ih}$. See **Figure 20.**

---

* damage a rogue program can cause

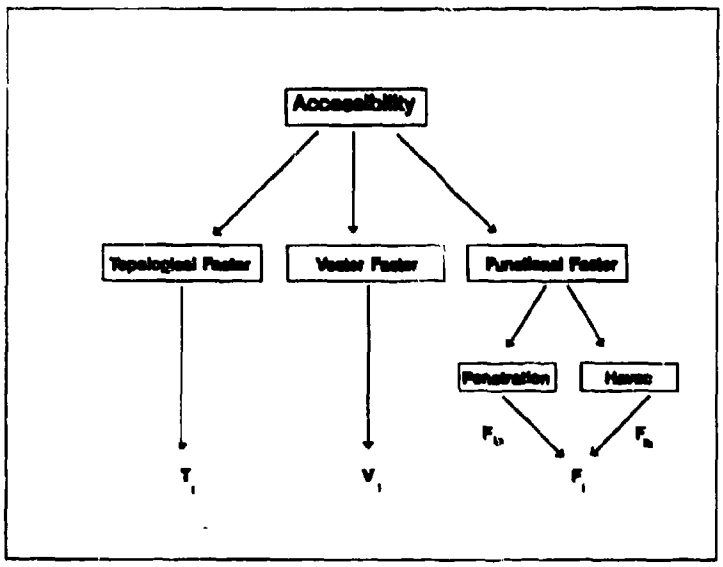


Figure 20. Accessibility Factors

Therefore, to calculate the VL:

$$VL = (V/T) * [\sum_{i=1}^{n} (F_i + \sum_{j=0}^{v} (F_i * F_j))]$$

Where:

T = topological factor
V = vector factor
F_i = functional factor or $F_{id} * F_{ih}$
F_{id}= penetration likelihood
F_{ih}= havoc/damage likelihood
i = index
j = index
n = number of subsystems (nodes)

3.4.3.2 Yearly Cost of Safeguards (YCSG)

To determine if the costs of applying safeguards are justified, the YCSG must be computed. To determine the yearly cost for cryptographic equipment for encrypting, digital signatures and authenticating messages, and the various antirogue products such as scanners, eradicators, monitors and cryptographic checksums, the following subset of Fred Cohen's[107] parameters are used:

1. the number of scans/checks (C) to be conducted
2. the loss of employee productivity (P) during scans
3. the time (T) to perform the scan
4. the one-time cost for licensing/purchasing (L) the product
5. the cost for key management (M)
6. the cost for installation and updates (U), which includes the time to install/update (U_t) plus labor costs U_e, and
7. the cost for eradicating (E) detected rogue programs, which includes the time required to clean up (E_c), to restore damaged files (E_r), and costs of labor (E_e).

Therefore, the yearly cost of safeguards is calculated as follows:

Yearly Cost of SafeGuard (YCSG) = (CPT + M + U + L + E)

Where $U = U_i + U_e$

$E = E_r + E_e + E_c$

3.4.3.3 Basic and Recurring Costs

To determine the one time basic cost and the recurring communication cost of each system without any incorporated safeguards, the following subset of Linda Rutledge's[208] parameters are used:

1. basic costs (BC) (nonrecurring) including the cost for:
   * hardware (N_h),
   * software (N_s)
   * installation (N_i) and
   * network connection (N_n)
2. recurring costs (RC), including costs for:
   * call initiation cost (R_b), which consists of:
     ** overhead cost to establish communication with the destination computing system (R_{bo}), plus
     ** cost of the time that a carrier signal

must be present for the destination computing system to respond (R_{br}), times

** the number of ports (R_{bp}), and

* the cost for data transmission overhead (R_c)

Therefore, the **Total Cost** = S(BC + RC)

Where:

S = the number of subsystems

Basic Cost (BC) = N

Where N = Nh + Ns + Ni + Nn

Recurring Cost (RC) = R

Where R = Rb + Rt

Rb = Rbp(Rbo+ Rbr)

At this point, we can calculate the ratio of the expected loss if no safeguards are implemented and the expected loss if safeguards are implemented as follows:

Loss-Ratio=(VL*(BC+RC)) / ((VL*((1-%SAFE)*(BC+RC)))+CSG)

Where :

VL*(BC+RC)=the expected loss if no safeguards are implemented

((VL*((1-%SAFE)*(BC+RC)))+CSG)=the expected loss with safeguards implemented

%SAFE=the percentage of protection the safeguards provide

Therefore, the **Cost-Benefit Ratio** for an unprotected system is calculated as follows:

Cost-Benefit Ratio = BSMG / CSFG

Where:

BSMG = Benefit per system from safeguards
= (VL*BC) - (VL*((1 - %SAFE)BC))

CSFG = Cost per system for safeguard
= CSG

This completes the abstract model. Section 4.5 instantiates the cost-benefit analysis using 3 computing systems consisting of a total of 9 links to determine if the defensive measures are cost-justified.

3.5 Summary

This chapter developed a generic abstract model of the rogue code insertion process into a communication channel via RF, delineated the parameters, requirements and resources to insert the code and examined insertion goals and methods. Moreover, the proposed cost-benefit component was discussed to determine whether it would cost more to implement a control or to accept the anticipated cost of the loss. The cost-benefit analysis allows the user to determine the point of diminishing returns whenever the benefit per system, which is the expected loss without safeguard minus the expected loss with safeguards, equals the cost of safeguards.

Building on the work of Cohen[109] and Rutledge[110] to determine the total costs of computing systems used for transmitting messages, a cost-benefit component to determine the cost effectiveness of using defense controls against rogue programs was proposed. The next chapter instantiates the abstract model on a DOS-based computing system using O'Neill Communication wireless LAN, called LAWN, to insert rogue code into a targeted host by RF.

Chapter 4 MODEL INSTANTIATION

4.1 Introduction

This chapter instantiates the abstract model developed in chapter 3 to insert rogue code into a target host's communication data stream using RF and provides data for the formulas developed in chapter 3. The technique used to instantiate the abstract model presumes that an adversary intent on inserting rogue code can covertly monitor all communications traffic among legitimate network members. The chapter also discusses how the model matches the conditions of the abstract model, examines the two control mechanisms implemented to prevent rogue code insertion via RF, provides examples of the cost-benefit component proposed in chapter 3 to determine if the defensive measures are cost-justified, and discusses various techniques to insert rogue code into a targeted host.

Chapter 4 has seven sections: the introduction (4.1); experiment setting (4.2); abstract model instantiation including the environment description to instantiate the model on a DOS-based system (4.3); two control mechanisms implemented to prevent rogue code insertions via RF (4.4) and cost-benefit analysis that determines that the defense measures were cost-justified (4.5); techniques to insert rogue code into a targeted host (4.6); summary (4.7).

4.2 Background

The chapter describes the Remote Insertion of Rogue Code (RIRC) Experiment conducted on 18 October 1991. The author uses the ISO 8473 Connectionless-mode Network Protocol[111] (CLNP) for network level connectivity, and the Trivial File Transfer Protocol[112] (TFTP) as its basic transmission protocol through applications level connectivity to insert the rogue code. CLNP and TFTP perform the same functions as the better known communication protocols, the Transmission Control Protocol[113] (TCP) for network level connectivity and the Internet Protocol[114] (IP) for its transmission protocol, respectively.

In the RIRC experiment, three IBM PC-compatible computer systems connected an RF local area network, using O'Neill Communications' Local Area Wireless Network (LAWN) modules. The author copied a file between two computer systems to ensure that the file transfer software worked. The third computer system, acting as an imposter, was then activated to insert rogue code into the data stream as the file was being transferred a second time between the same two legitimate computer systems. The innocent recipient computer executed the rogue code when it executed the infected file, thereby illustrating the rogue code's successful insertion.

The experiment showed that a rogue program could be inserted via RF into a network with only built in security mechanisms. Although intruders have more difficulty subverting secure networks, the above experiment is valid although insertion techniques are more complex for these systems.

4.3 Parameters and Requirements

Determining the target host's parameters and requirements was not difficult. In the experiment, because the intruder host used the same hardware module as the bonafide hosts, he knew the target network's communications protocol. The LAWN module automatically formatted the rogue code and the receiving host accepted the formatted code. The following three sections discuss the communications channel, data stream conformation, and code generation for this instantiation.

4.3.1 Communications Channel

The experiment used the seven-layer ISO OSI reference model, the ISO 8473 Connectionless-mode Network Protocol[115] (CLNP) and the Trivial File Transfer Protocol[116] (TFTP).

4.3.1.1 Connectionless-mode Network Protocol (CLNP)

The I3O 8473 CLNP provides network-level connectivity. Residing at level 4, the transport layer of the OSI seven-layer model[117] provides end-to-end communications. The latest international standard network protocol, the Government Open System Profile (GOSIP)[118] mandates CLNP. CLNP identifies and categorizes the method to perform functions within the network layer, provides a uniform structure and describes which protocols provide the OSI network service.

4.3.1.2 The Trivial File Transfer Protocol (TFTP)

TFTP is a small, easily implemented protocol that transfers files at the application level[119]. For example, some diskless UNIX client machines use TFTP to load their operating system[120]. Diskless workstation manufacturers can place TFTP in many platforms read-only memory (ROM) to bootstrap the system when the machine is on. TFTP's advantage is that it allows bootstrapping code to use the same protocols as the running systems[121]. Its features are limited to reading and writing files from a remote server. Any transfer activates a

request for connection to read or write a file. If the server authorizes the request, the connection is opened and the file is sent in 512 byte packets.

4.3.2 Data Stream Conformation

The target host accepts the rogue code as normal network data because the LAWN module had formatted it properly for CLNP and TFTP.

4.3.2.1 CLNP Data Stream

The two CLNP protocol data units (PDUs) that transfer data and report errors are the data protocol and error report PDUs. DPUs contain octets (bytes) that are numbered sequentially starting with number one. When a data PDU is discarded, an error report PDU is generated which identifies the PDU that was discarded, why it was discarded, and where the error occurred. Both PDUs have five parts[122]:

1. the fixed part
2. the address part
3. optional segmentation information part
4. optional switches part
5. optional data part.

Figure 21 shows a data PDU's structure.

An error-report PDU's structure is not shown.

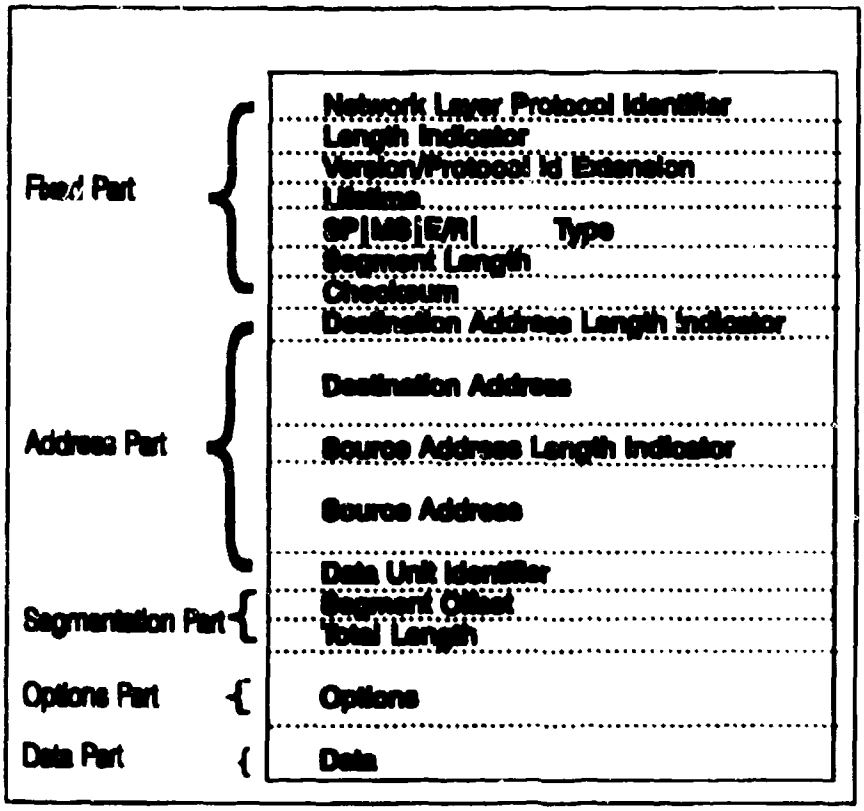


Figure 21. Data Protocol Data Unit (PDU) Structure

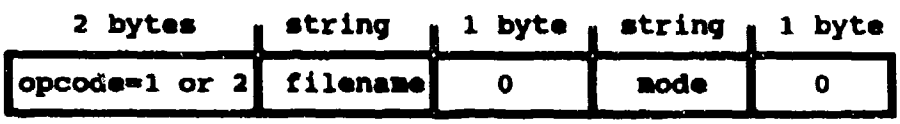
Both PDUs are padded to an integral number of octets and each data octet is numbered. To avoid duplicating data between sessions, each session's first octet is assigned a unique number for the virtual connection (VC). This sequential number starts with one. Other packets are assigned numbers incrementally as they are transferred. These unique numbers assure the receiving host that the data is legitimate and is arriving in order.

4.3.2.2 TFTP Data Stream

The TFTP packet contains one of the following five opcode headers:

1. Read Request (RRQ) = 1
2. Write Request (WRQ) = 2
3. Data (DATA) = 3
4. Acknowledgment (ACK) = 4
5. Error (ERROR) = 5

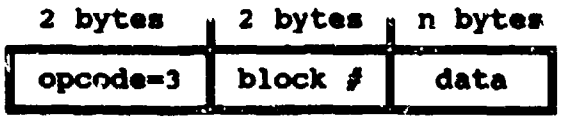
The Read Request/Write Request packets have the following format:



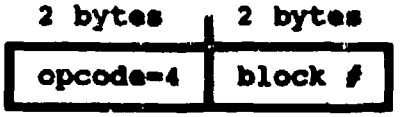
The filename and the mode string are zero-terminated ASCII

characters. TFTP supports three transfer modes: ASCII (8 bits), binary (8 bit bytes), and mail which allows it to be integrated with electronic mail.

The data packet has a block number and a data field. The block number starts at 1 and increases sequentially by one for each additional packet. The data field is from 0 to 512 bytes long. The data packet format follows:



The acknowledgment packet acknowledges all but termination and timeout packets. The receiver must acknowledge each packet individually block #. The acknowledgment packet format follows:



The error packet contains an integer which indicates the error type:

0 = <not defined>

1 = <files not found>

2 = <access violation>

3 = <disk full or allocation exceeded>

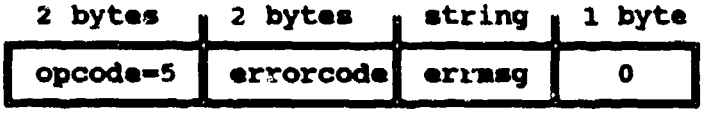
4 = <illegal TFTP operation>

5 = <unknown transfer ID>

6 = <file already exists>

7 = <no such user>

The error message, like all the other strings consists of zero-terminated ASCII characters that explains the error's nature to the user. The error packet format follows:



4.3.2.3 Coding Characteristics and Synchronization

To ensure that the rogue code "looks-like" the code it is replacing, CLNP and TFTP code characteristics are coordinated with their synchronization.

CLNP

CLNP packets contain 512 octets. Synchronization is every 500 ms, and priority codes handle contention. The priority parameter's value indicates the relative priority of the PDU. Priorities vary from 0 (the default) through 14 (the highest).

A checksum octet, applied at the source node and authenticated at the destination node, assures data integrity. The checksum

is computed on the entire PDU header, which includes the segmentation* and options information if available for a data PDU. For an error-report PDU, checksum includes the reason for discard as well.

CLNP requires positive acknowledgement for all of the data it sends. If the destination or receiver does not acknowledge data integrity within a specified timeout period, the sender will retransmit the data. The sender retransmits the data for some number of iterations before it resets the connection. The length of the timeout period is based on packet size of 512 octets, specified in increments of 500 ms. For example, the timeout period is 500 ms for each packet with five retries[123]. The receiver discards duplicate packets.

TFTP

The size of a TFTP packet is 512 octets. Synchronization is every 500 ms. Each TFTP data packet is assigned a block number which is assigned consecutively starting with one. Each data packet contains one data block which must be acknowledged with an acknowledgment packet before the next packet is sent. If a packet gets lost enroute, the sender can

---

* Used when the size of the PDU is greater than 512 octets.

transmit the packet for a set timeout period of 3 seconds. After 3 seconds, the connection is terminated. The connection is reset also after a preset number of retries. The receiver discards all the duplicate packets.

4.3.2.4 Transmission Frequency

Determining the transmission frequency was unnecessary because the intruder successfully inserted the rogue code message as the first packet.

4.3.3 Experiment Resources

Analogous to the required hardware for the generic abstract model discussed in chapter 3, the experiment resources consisted of a DOS-based computer system and the LAWN module which contained a microprocessor and a radio transceiver that sent and received data via radio signal. The module served the same purpose as the CHRIS and the transceiver from the abstract model. The configuration was comparable to a LAN and can be adapted to a WAN using repeaters or more powerful transceivers. The workstations were connected via RF modems to provide the physical and link-level connectivity.

For purposes of this experiment, the two authorized hosts are named Aaron and Bill and the unauthorized user is named Intruder.

Figure 22 provides a system overview of the network.

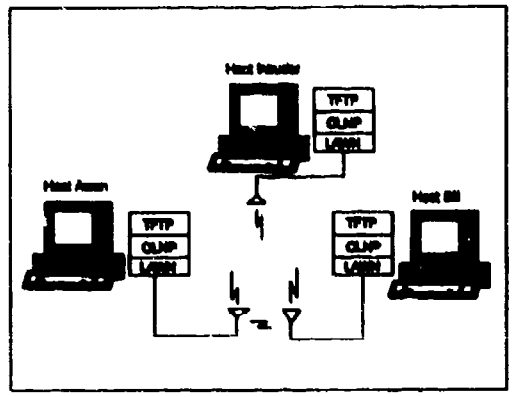


Figure 32. Hardware System Overview

The three system's hardware configuration follows.

1. For the computing system

* Host Bill is a Packard Bell IBM-compatible computer system with a 12 MHz Intel 80286 CPU, with two 5 1/4 inch floppy disk drives, 640K of RAM and a VGA monitor.
* Host Aaron is a BragL IBM-compatible computer system with a 25 MHz Intel 80386 CPU, with a 5 1/4 inch high density floppy disk drive, 3.5 inch high

density floppy drive, 80 Mbyte hard disk, 640K of RAM and a super VGA monitor.

* Host Intruder is a Zenith-150 IBM-compatible computer system with a 4.77 MHz Intel 8088 CPU, with two 5 1/4 inch floppy disk drives, 640K of RAM and a CGA monitor.

2. The LAWN Module connects the three computers to the network wirelessly, using high-frequency spread spectrum radio transmissions which distribute the transmitted data across multiple frequencies[124].

Spread spectrum uses a pseudorandom sequence generator by adding from 10 - 1,000 bits to the signal. Spreading the bits results in a new signal[125] which is distributed over a wide range of frequencies for transmission. This signal is then reduced to the size of the original frequency at the receiver[126]. See Figure 23 for LAWN specifications.

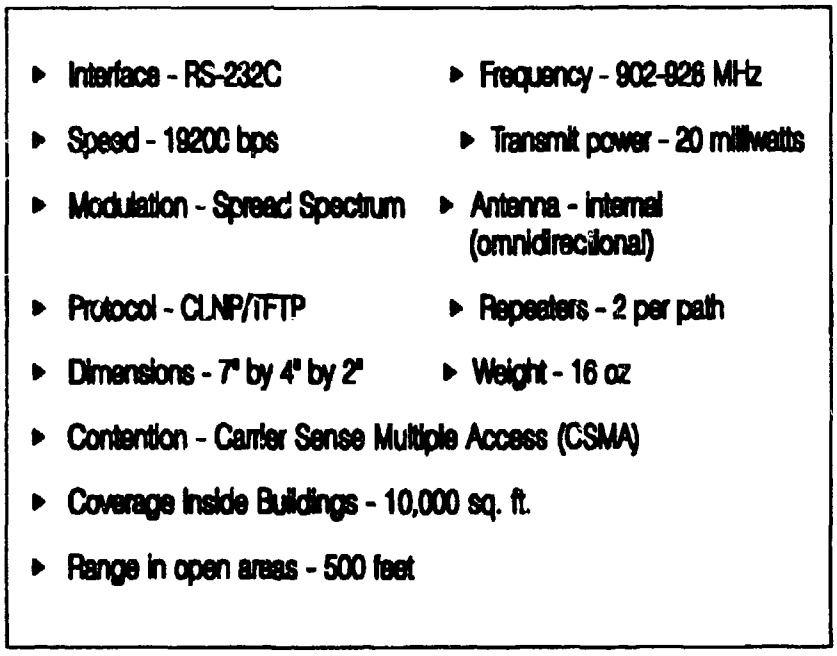


Figure 23. Lawn Specifications

It is easy to install the LAWN module. For example, it weighs 16 ounces, is six inches long and two inches wide. It includes all the software necessary for electronic mail, file transfers and peripheral sharing as well as AC power adapters and 9- and 25-pin RS-232 serial port connectors. The module is easy to install, easy to use, and easy to move. When the user plugs in the module into the serial port of the computer, the power source executes its software.

The module has four lights on the front panel which indicate the LAWN's status (**Figure 24**). A summary of the four indicator lights follows:

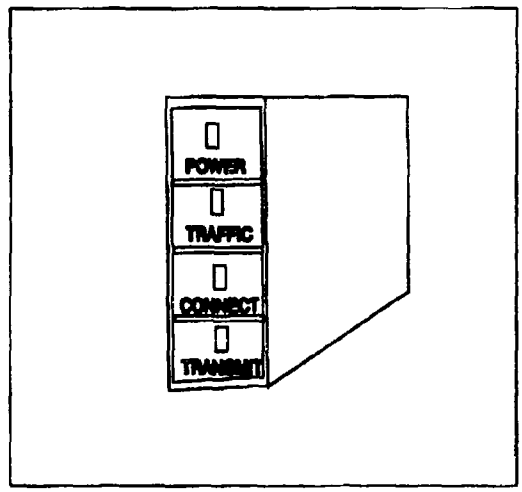


Figure 24. LAWN Schematic

1. The red POWER light indicates the module is receiving power. This light blinks when the module is receiving a message.
2. The green TRAFFIC light signifies that the module is in use.
3. The green CONNECTED light indicates that the computer is communicating with another machine.
4. The green TRANSMIT light indicates that the computer is sending data to another machine.

4.3.4 Code Generation

Because the time period when the intruder can inject code during a transmission is limited, he used a packet size rogue code of 512 bytes and followed this three step methodology:

1. Initialized the hosts to transfer files.
2. Executed a normal file transfer.
3. Inserted the rogue code during file transfer.

4.3.4.1 Initialising Hosts to Transfer Files

The author initialized the three computer systems, host Aaron, host Bill, and host Intruder, by connecting the LAWN modules to each system via the RS-232 serial port connectors. He inserted two 5 1/4" diskettes in each system's drives A and B and typed the <start> command on the command line in drive A to initialize each host. Initialization occurs when the applicable software programs are executed.

Initialization ensures that the system is set up to perform its function, such as identifying each specific host on the LAN, ensuring that the peripheral device controlled by the driver is present and functional, and processing the communications between the application and the computer LAWN. Initialization consists of the following four steps:

1. Set hostname
2. Assign packet drivers
3. Initialize CLNP network layer software
4. Initialize TFTP software

The system must be initialized for file transfer. See Appendix 1 for the batch code for this initialization.

4.3.4.1.1 Set Hostname

First, the author assigns each computer system a hostname so that the network can uniquely identify each system. The commands to set this parameter are:

1. For host Aaron ==> SET HOSTNAME=Aaron
2. For host Bill ==> SET HOSTNAME=Bill
3. For host Intruder ==> SET HOSTNAME=Aaron
   (the imposter host is masquerading as host Aaron)

4.3.4.1.2 Assign Packet Drivers

Then, to provide the link layer connectivity, packet drivers for each host were assigned in accordance with each machine's specific hardware configuration as discussed in section 4.3.3.

For each specific host, the following parameters* were used:

1. Host Aaron ==> For COM1: lawnslip 0x65 -h 6 3 0x2f8 19200

2. Host Bill ==> For COM3: lawnslip 0x65 -h 6 4 0x3f8 19200
(COM1 and COM2 were already being used)

3. Host Intruder==> For COM1: lawnslip 0x65 -h 6 4 0x3f8 19200

4.3.4.1.3 Initialise CLNP Network Layer Software

Second, the author initialized the CLNP network layer software for each system by executing the command <clnptsr>. The CLNP software is a TSR memory resident program that provides telecommunications and information exchange between systems. See citation[137] for the CLNP code.

4.3.4.1.4 Initialise TFTP Software

Thirdly, the TFTP software, which contains both server and client processes is automatically initializes when it executes a file transfer beginning with the command <tftp> for a

---

* Usage:
LAWNSLIP [-n] [-d] [-w] packet_int_no [-h] [-p count] [-t count] [driver_class] [int_no] [io_addr] [baud_rate] [send_buf_size] [recv_buf_size] [data_buf_size]
-h enables hardware handshaking
-p modifies limit before polling mode used
-t modifies the timeout for dallying after last character.
The driver_class could be SLIP, KISS, AX.25, or a number.

bonafide host or <bftp> (bad file transfer protocol) for an intruder host. **BFTP** is a modified version of TFTP that allows the host intruder to monitor all traffic and insert rogue code in the first packet sent to the receiver host. This completes the hosts' initialization process.

4.3.4.2 Executing a Normal File Transfer

To ensure that the RF modems were operational and that the two friendly hosts could communicate, the author sent a file from host Aaron to host Bill: he enters a "request wait" command at host Aaron by invoking <tftp> as follows:

Typed from host Aaron: <tftp>

At Host Bill the author requests the file "crc.exe" from host Aaron and renames it "test.exe", as follows:

Typed from host Bill: <tftp -h Aaron -g crc.exe test.exe>*

The TFTP specifies that the TFTP protocol is to be used to communicate between hosts Aaron and Bill. The second and third parameters, -h Aaron, specify Aaron as the source address. The remaining parameters request the file <crc.exe> be transferred from host Aaron to host Bill and renamed <test.exe>. To verify that test.exe is an exact duplicate of crc.exe, the author conducted the following three tests: a file size test using the DIR command, a CRC, and a byte-by-byte comparison. The "DIR" command shows that the files have the same size - 3273 bytes. A CRC via crc.exe established that the CRC values for both files were identical - 5E A2 for crc.exe and test.exe.

Where:

---

* Usage:
Without -h, -p or -q, Server Operation
Client Operation must supply either -h hostname or -a address, with
-p local_filename remote_filename to put a remote file or
-g remote_filename local_filename to get a remote file.

[Option parameters with (default settings)] as follows:
[-r (5)] Retry attempts before giving up
[-s (512)] PDU data size
[-u (69)] TFTP protocol selector #
[-f (1)] Fragmented PDU's Permitted, No = 0, Yes = 1
[-c (0)] Header Checksum Requested, No = 0, Yes = 1
[-e (1)] Error Reports Requested, No = 0, Yes = 1
[-d (0)] Debug level, 0=none, 1=some, 2=detailed

crc.exe is the file that was sent by host Bill

test.exe is the renamed file crc.exe

In addition to the CRC check, although there was a 1 to 2^{16} chance that two files will have the same CRC value[128] (using four characters), the author conducted a byte-by-byte comparison using the DOS COMPARE command for further verification as follows:

```
A:\>comp crc.exe test.exe
Comparing CRC.EXE and TEST.EXE.....
   Files Compare OK
```

The COMPARE command showed that crc.exe and test.exe files were the same. Therefore, the author could successfully transfer files between the two friendly computer systems.

To understand how to insert rogue code during a file transfer, it is beneficial to examine the data flow between the machines during file transfer. First, host Bill sends a read request for the file "crc.exe" from host Aaron. Host Aaron opens the file and reads the first block of 512 bytes into a buffer. A PDU is then created using the addressing information in host Bill's read request. A sequence number of 1 is assigned to the first data block which is sent to Host Bill (Figure 25).

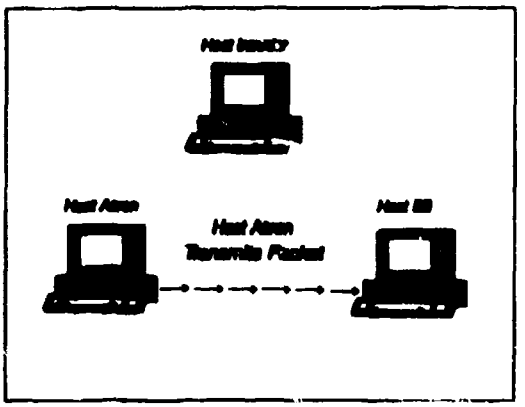


Figure 25. Host Aaron Sends a Message to Host Bill

Upon successful receipt of the data, host Bill sends an ACK PDU with the same sequence number, seq. #1, to host Aaron (Figure 26).

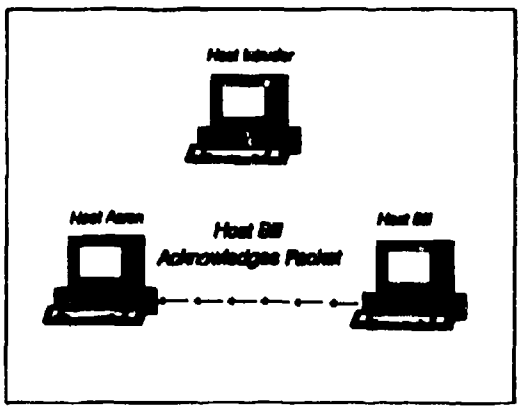


Figure 26. Host Bill Acknowledges Host Aaron's Message

Host Aaron, upon receiving the ACK, continues to send blocked packets with sequentially increasing sequence numbers until the entire file is transferred. If an ACK is not received within a specific time, the packet is retransmitted until either an ACK is received or a timeout has been reached. If the packet is not in the correct format or the checksum in the network protocol or a CRC in the LAWN protocol does not match, the packet is rejected.

4.3.4.3 Inserting Rogue Code During a File Transfer

At this time, the intruder inserts rogue code into the friendly host's data stream. To invoke the protocol, the imposter, masquerading as host Aaron, with the HOSTNAME=Aaron, executes <bftp>. This command places the imposter host in a monitoring mode, ready to insert its code as soon as it detects a file transfer. No operator interaction is required for this process. To insert the rogue program, host intruder creates a spurious PDU whose format is identical to the legitimate system's PDU format; the spurious PDU must pass the CLNP network layer checksum, pass the link layer CRC test, and have the same sequence number and format as the good packet. The intruder uses the same procedures as set forth in paragraph 4.3.4.2 to effect a normal file transfer. As host Intruder detects a file transfer taking place, it immediately sends its "spurious" packet to the receiver host, host Bill

(Figure 27).

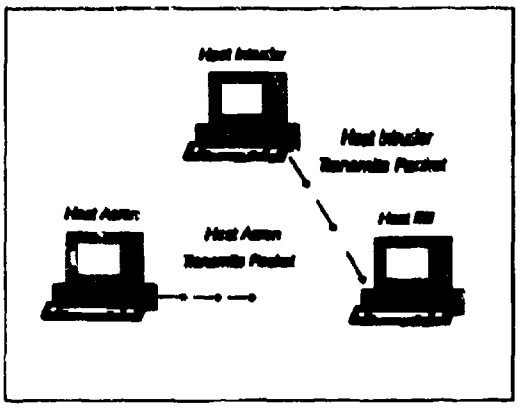


Figure 27. Host Intruder's Packet Reaches Host Bill First

Host Bill accepts the bad packet and sends an ACK to host Aaron indicating that the first packet has been successfully received (Figure 28).

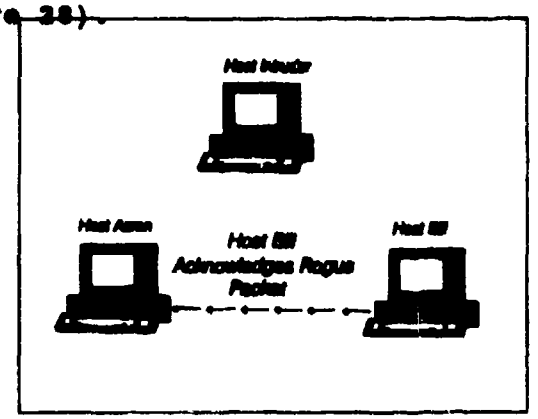


Figure 28. Host Bill Acknowledges Host Intruder's Packet to Host Aaron

The above operational steps took place in this order:

Step	Timeline
0	Host Bill sends file request to host Aaron.
1	Host Intruder detects that a file transfer is to take place.
2	Host Intruder sends its prepared rogue packet, spurious packet #1 to the receiver host, host Bill.
3	Host Aaron prepares its message for host Bill and sends its first packet bonafide packet #1.
4	Host Bill receives rogue packet, spurious packet #1 from host Intruder.
5	Host Bill receives host Aaron's packet, bonafide packet #1 and discards it, because has already received packet #1. No ACK is sent for rejected packets.
6	Host Bill acknowledges receiving packet #1 (really sent by host Intruder) to host Aaron
7	Host Aaron receives acknowledgement for (rogue) packet #1 (sent by host Intruder), and then continues to send the other packets

Host Intruder will almost always beat the sender host because the sender host has much more to do than the host Intruder to prepare a packet for transmission such as finding and opening the file and preparing and sending PDUs. To illustrate this point, the host Intruder was the slowest machine with a 4.77 MHz CPU clock speed; the sender, host Bill, was the fastest computer with a 25 MHz CPU clock speed.

The following paragraphs describe why the intruder's packet got to host Bill before the sender's packet.

DOS programs use a unique, 16-bit value called a file handle to perform file operations[139]. The file handle identifies the file currently being accessed and the operation to be performed, such as to open or create files and subsequent functions to perform other file operations such as reading and writing. The following describes the timeline and steps required to transfer files:

Step	Timeline
0	Receiver host Bill requests a file from the sender host Aaron.
1	Sender host must first locate the requested file via the **find first file** function, Interrupt 21, Function 4EH.
2	The sender host opens the found file via the **open file** function, Interrupt 21, Function 3DH.
3	Sender host places the file in an internal buffer to prepare it for transmission and to prepare the data PDU.
4	Sender host transmits the data PDU to the receiver host.

The host Intruder does not have to follow these steps with its rogue code packet already prepared, as soon as it detected a file transfer operation request, it immediately transmitted

its prepared packet to the receiving host. The intruder can, therefore, prepare the rogue packet in advance and skip the file I/O.

An examination of test.exe demonstrates that the insertion was successful. Performing a DIR command, the file size is the same as the original file, crc.exe: 3273 bytes. But, a CRC check shows that the CRC value is different, BD 9F. Also, a byte-by-byte comparison using the DOS COMPARE command shows:

```
A:\>comp crc.exe test.exe
Comparing CRC.EXE and TEST.EXE...
Compare error at OFFSET 0
     file1 = 4D
     file2 = BB
Compare error at OFFSET 1
     file1 = 5A
     file2 = 2
Compare error at OFFSET 2
     file1 = C9
     file2 = 0
Compare error at OFFSET 3
     file1 = 0
     file2 = B9
Compare error at OFFSET 4
     file1 = 7
     file2 = 40
Compare error at OFFSET 5
     file1 = 1
     file2 = 8C
Compare error at OFFSET 6
     file1 = 0
     file2 = CA
Compare error at OFFSET 7
     file1 = 20
     file2 = 8E
Compare error at OFFSET 8
     file1 = 0
     file2 = DA
Compare error at OFFSET 9
     file1 = 0
     file2 = BA
10 Mismatches - ending compare
```

The COMPARE command shows that the two files crc.exe and test.exe are different only at the program's first ten bytes - where the rogue code was inserted.

When host Bill executes the infected test.exe, it displays the message, "This file has been infected with a harmless computer virus! This file is no longer good".

The "bftp" software notifies the rogue operator that the rogue program was transmitted as well as how many packets the sender host transferred. See **Appendix 2** for the transmitted rogue program's code.

4.3.4.4 Experiment Summary

The above three sections describe how a rogue program is inserted into a wireless communication stream. The imposter, host Intruder, masqueraded as the sender host, host Aaron, by creating packets that look like they came from host Aaron. The imposter monitored all traffic between the two friendly hosts, Aaron and Bill. Once the imposter detected that a file transfer was to take place, it immediately forwarded its spurious rogue code packet to the receiver host Bill, which acknowledged to the sender host Aaron that the packet was the

bonafide sender's first packet. Host Bill, upon receiving host Aaron's legitimate first packet, discarded it as a duplicate. Host Aaron, upon receiving an acknowledgement for its "supposed" first packet, continued to send the rest of the packets. Therefore, host Intruder was able to insert successfully its rogue code into the file that host Aaron sent to host Bill. The next section discusses the defense measures that hosts Aaron's and Bill's users could have taken to minimize the host Intruder's threat.

4.4 Defense Measures

For purposes of this dissertation, only the first two defense measures of the seven that Chapter 3 discussed, CRC and checksum, were used in the experiment to demonstrate successful detection of rogue code insertion. The DIR command showed that intruder modified the original file, crc.exe because the infected file, test.exe, was not the same size as the original file. The checksum, COMPARE command, reinforced the fact that the two files were not the same via a byte-by-byte comparison. The CRC clearly showed that the two files were different lengths.

4.5 Cost-Benefit Analysis

In this section, the formulas provided in chapter 2 are implemented with examples to determine whether it would cost more to implement controls or to accept the anticipated cost of the loss.

To implement the cost-benefit portion of the model to ascertain the cost-benefit ratio, the following three parameters must be computed:

1. the accessibility of computing systems to rogue program attacks, access vulnerability likelihood (VL)
2. the cost of applying antirogue products ((the yearly cost of safeguards (CSG) which enhances product effectiveness))
3. the basic cost (BC), recurring cost (RC) and the expected yearly loss of the computing system

4.5.1 Access Vulnerability Likelihood (VL)

Using the formula from page 75,

$$VL = (V/T) * [\sum_{i=1}^{n} (F_i + \sum_{j=0}^{v} (F_i * F_j))]$$

Where:

T = topological factor
V = vector factor
F_i = functional factor or F_{id} * F_{ih}
F_{id}= penetration likelihood
F_{ih}= havoc/damage likelihood
i = index
j = index
n = number of subsystems (nodes) that can be carriers

three computing systems were used consisting of 9 links as shown in **Figure 29**. The topological factor is 9, since there is a total of nine links.

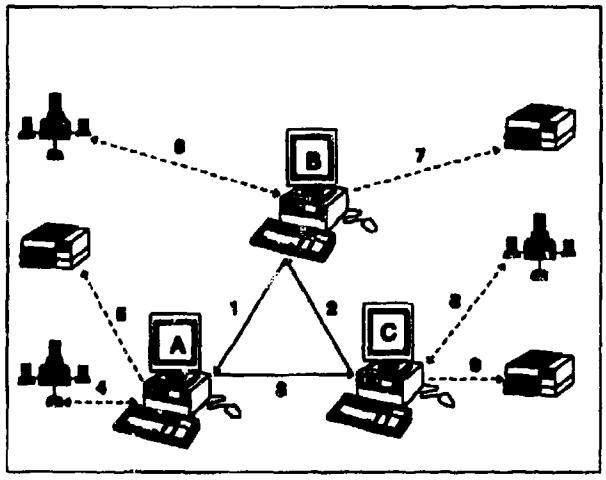


Figure 29. Example Subsystem

The vulnerability, based on the vector analysis contribution was .66 because 6 of the 9 links can carry the infection.

The likelihood of the three printer links becoming carriers was remote. Dividing the vector contribution by the topological contribution and multiplying it by the function contribution of each of our subsystems determined the VL. The function contribution was determined from the following matrix which contains a number of safeguards with associated (subjective) weights assigned specifically for purposes of this dissertation (other researchers may assign different weights depending on their own experiences or purposes):

Functional Factor Matrix

USE OF SAFEGUARDS	WEIGHTED VULNERABILITIES	
	Penetration (F_{id})	Damage (F_{ih})
1. CRCs	.50	.50
2. checksums	.30	.30
3. encryption	.50	.10
4. digital signatures	.10	.99
5. incorporated safeguards	.30	.99
6. SW or HW mechanisms	.10	.10
7. combination of the above	.10	.10

For example, in the case where 3 computing systems had a total of 9 links, 6 of which can be carriers, assuming that computing system-A is using software or hardware safeguards with its respective weight where $F_i = (F_{id}) * (F_{ih}) = .10 * .10 = .01$, and computing system-B is using CRCs where $F_i = .50 * .50 = .25$ and computing system-C is using no safeguards, $F_i = .99$, to determine VL (see **Figure 30**), one calculated:

For n=1: VL(1)=.66[.01+(.01*.25)+(.01*.99)] = .015

For n=2: VL(2)=.66[.25+(.25*.01)+(.25*.99)] = .330

For n=3: VL(3)=.66[.99+(.99*.01)+(.99*.25)] = .825

Therefore, VL = VL(1)+VL(2)+VL(3) = 1.17

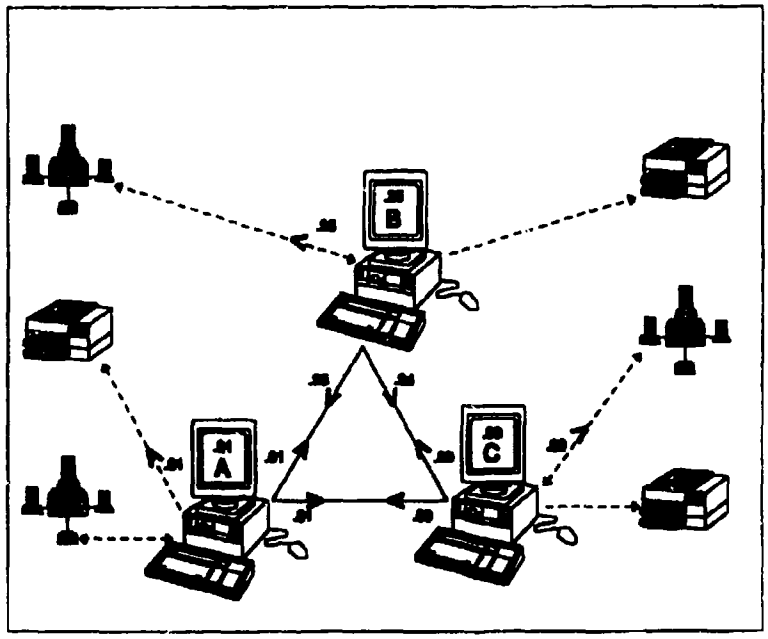


Figure 30. Computing Systems Vulnerability Likelihood

Hence, the VL for the three computing systems is 1.17, meaning that there is a lesser chance than the mean* that the network may be infected. Converting VL to a percentage for use in forthcoming calculations, 1.17 approximates to a 20% vulnerability. This percentage is determined by calculating the lower bound (i.e., full protection which is defined at 99%) and the upper bound (no protection at .01%) vulnerability for this network, and then normalizing the upper bound. For the above three computing systems, the lower bound is

For n=1: VL(1)=.66[.01+(.01*.01)+(.01*.01)] = .0067

For n=2: VL(2)=.66[.01+(.01*.01)+(.01*.01)] = .0067

For n=3: VL(3)=.66[..01(.01*.01)+(.01*.01)] = .0067

Therefore, VL = VL(1)+VL(2)+VL(3) = .0202

The upper bound vulnerability is

For n=1: VL(1)=.66[.99+(.99*.99)+(.99*.99)] = 1.947

For n=2: VL(2)=.66[.99+(.99*.99)+(.99*.99)] = 1.947

For n=3: VL(3)=.66[.99+(.99*.99)+(.99*.99)] = 1.947

Therefore, VL = VL(1)+VL(2)+VL(3) = 5.841

Therefore, the percentage equivalent of VL is determined by normalizing the lower bound to 1, such that 1.17/5.841 = 20%. Hence the accessibility vulnerability likelihood is 20%. See the following table for an analysis of VL as the value of n doubles, while keeping all other parameters constant.

---

* The mean is determined by averaging the VL for the three subsystems with no vulnerabilities (i.e, F(i)=.01; hence, VL=.0202) and with full vulnerabilities (i.e., F(i)=.99; hence, VL=5.841). Therefore, the mean=2.93.

N	VL	LB	UB	%
3	1.17	.0202	5.841	20
6	2.34	.0404	11.682	20
12	4.68	.0808	23.364	20
24	9.36	.1616	46.728	20
48	18.72	.3232	93.456	20
96	37.44	.6464	186.912	20

Where:
N = # of nodes
VL = Measure of Vulnerability
LB = Lower Bound
UB = Upper Bound
% = Normalized Value

As expected, the normalized VL does not change when n is doubled and all other factors remain the same. Hence, adding more nodes to a network does not change the percentage of the VL as long as all the other parameters remain the same. There is no difference in the VL percentage when adding or subtracting nodes when all other parameters are constant.

4.5.2 Yearly Cost of Safeguards

To determine the yearly cost of safeguards, the following equation was used:

Yearly Cost of SafeGuard (YCSG) = (CPT + M + U + L + E)

Where:

C = the number of scans/checks

P = the loss of employees' productivity

T = the time to perform the scan

M = the cost for key management

U = the cost for installation and updates
= Ui (the time to install/update) + Ue (the employees' costs)

L = the cost for licensing/purchasing of product

E = the cost for eradicating detected rogue programs
= Ec (the time required to clean damaged files) + Er (time to restore damaged files) + Ee (employees' time involved)

For example, to calculate the safeguard costs, it was assumed that a cryptographic mechanism and a scanner/eradicator were installed such as the RSA algorithm and SCANV that provided 99% protection, as per the **Functional Factor Matrix.**

COST FOR SAFEGUARDS FOR COMPUTER SYSTEMS:

Number of Scans (C):	250	(Scan done after each bootup - daily)
Loss of Employee Time (F):	.27	(@ $16.00 per hour = $.27/min)
Time to Perform Scan (T):	5 min	(for 80 MByte HD, 25 MHz)
Key Management Costs (M):	$50.00	(one time cost of equipment)
Cost for Installation (W):	$24.00	(one hour for crypto, 1/2 hour for scan)
Cost for Updates:	$40.00	(4 updates per year @ $10.00 per scan)
Cost for Employee:	$32.00	
Cost for Licensing (L):	$25.00	(per year)
Cost to Eradicate (E):	$.025	(scanner will automatically eradicate w/permission - approx 5 sec)
Cost to Restore:	$.54	(assume back-ups available - 2 min to get them)
Cost of Employee:	$4.00	(assume no major damage - 15 min to back-up specific files)

Yearly Cost of Safeguards = (250*.27*5+50+96+25+4.55)

= $513.05

Hence, the yearly cost of safeguards was $513.05 per year for each subsystem, which may be a reasonable cost depending on the importance of the data to be protected. The above costs were obtained from citations[130,131].

4.5.3 Basic and Recurring Costs

To determine the total cost of each subsystem without any incorporated defensive controls:

Total Cost = S(BC + RC)

Where:

S = the number of subsystems

Basic Cost (BC) = M

Where M = Mh + Ms + Mi + Mn

Mh = hardware costs

Ms = software costs

Mi = installation costs

Mn = network connection costs

Recurring Cost (RC) = R

Where R = Rb + Rt

Rb = Rbp(Rbo+ Rbr) = initiation cost

Rbo = cost to establish communication

Rbr = cost of the response time

Rbp = number of ports

Rt = cost for data transmission

Example: Assuming 3 computing systems connected via RF as in the previous example:

NONRECURRING COST				RECURRING COST		
hardware +	software +	install +	connection	call initiation +		transmission overhead
$ 3000.00 +	5000.00 +	0.00 +	450.00	.025 +		.075

Basic Cost = $8458.00

Recurring Cost = $.10

Where: For Basic (Nonrecurring) Costs

hardware cost - $3000.00

software cost - $5000.00

install cost - one employee working at $16.00 per hour for 1/2 hour

- $8.00

connection cost - the cost of the wireless LAN components, such as the LANN

- $450.00

Therefore, Basic Cost = $N_h+N_s+N_i+N_n$ = $3000 + $5000 + $8 + $450

= $8458.00

Where: For Recurring Costs

call init cost - one employee working at $16.00 per hour for 2 seconds for overhead (Rbo-$.010) and for 3 seconds for respond (Rbr-$.015) (@ $.005 per second)

- $.025

transmit cost - one employee working at $16.00 per hour for 15 seconds for a 60K file transmitted at 4K bps

- $.075 (assuming one port (Rbp=1))

Therefore, Recurring Costs = $R = R_b + R_t = R_{bp}(R_{bo} + R_{br}) + R_t$

= 1($.010 + $.015) + $.075

= $.10

Hence, Basic Costs plus Recurring Costs = $8458.10 per subsystem. The total system cost = 3 * $8458.10) = $25,374.30.

At this point, the ratio of the expected loss with no safeguards to the expected loss with safeguards for each subsystem was calculated as follows:

Loss-Ratio=(VL*(BC+RC)) / ((VL*((1 - %SAFE)*(BC+RC)))+CSG)

Where :

VL*(BC+RC) = the expected loss if no safeguards are implemented

(VL*((1 - %SAFE)*(BC+RC))+CSG) = the expected loss with safeguards implemented

%SAFE = the percentage of protection provided by the safeguards (its effectiveness)

Therefore, the **Cost-Benefit Ratio** for an unprotected subsystem was calculated as follows:

Cost-Benefit Ratio = BSMG / CSFG

Where:

BSMG = Benefit per subsystem from safeguards
= the expected loss if no safeguards are implemented minus the expected loss with safeguards implemented
= (VL*BC) - (VL*((1 - %SAFE)BC))

CSFG = Cost per subsystem for safeguard
= CSG

Recalling that the VL = 20%, the expected cost of loss without safeguards to the expected cost of loss with safeguards is:

Expected Cost of Loss Without Safeguards = (.20*8458.10)

= $1691.62

Expected Cost of Loss With Safeguards = ((.20*((1-.99)*8458.10))+513.05)

= $529.97

Loss Ratio = (.20*8458.10) / ((.20*((1-.99)*8458.10))+513.05)

= 1691.62 / 529.97

= 3.2 / 1

= 3 / 1

The Cost-Benefit Ratio for an unprotected subsystem was calculated as follows:

Expected damage per subsystem due to access vulnerability likelihood of .20 (from above) is = $1691.62

(which is the expected damage per subsystem WITHOUT safeguards)

Expected damage per subsystem with safeguards (from above) is $529.97

Therefore:

Benefit per subsystem from safeguards (BSMG) = (Expected damage)-(Expected damage w/safeguards)

= 1691.62 - 529.97

= $1161.65

Cost per subsystem for safeguard (CSPG) = $513.05

Cost-Benefit Ratio (for each subsystem) = BSMG / CSPG

= (1161.65) / (513.05) = 2.3

Therefore, the Cost-Benefit Ratio = 2.3

= 2 / 1

These figures demonstrate that for every dollar spent on safeguarding an unguarded system, the user will avoid spending approximately 3 dollars (loss ratio = 3/1). For every dollar spent, the user will save 2 dollars (cost benefit ratio = 2/1). If the user does not select this wise choice, then according to the vulnerability estimate, he could end up spending an additional $1161.65 due to rogue program infection and associated losses.

4.6 Attack Methodology Variations

In the above experiment, the intruder inserted rogue code into a targeted host via RF by replacing the first valid packet with a rogue packet. Insertion can also be accomplished by replacing any packet in the data stream; however, it is more difficult to insert rogue code successfully into other data stream locations because the timing sequence and packet order become more important. After the first packet, the rogue code could accomplish many different tasks such as deleting files, modifying programs, capturing programs or propagating its rogue code to other computing systems. In all these cases, the code would be more complicated and would require more than two packets. Moreover, the intruder host can masquerade as the receiving host as well as the sending host, thereby having the capability to eavesdrop on transmissions. The intruder is

able to masquerade as the receiving host assuming that the intruder knows the hostname of the bonafide receiver host as well as using the same hardware with the appropriate settings. Acknowledgments from the intruder are not a concern because the protocol will discard any duplicate acknowledgements. Also, the rogue code can accomplish many other tasks, but the more tasks it pursues, the more rogue code required, the more chances of something going wrong and, hence, the more prone the rogue code is to detection.

4.7 Conclusions

This dissertation demonstrated that unprotected wireless LANs are more vulnerable to rogue program attack than traditional LANs. This vulnerability was demonstrated by developing and instantiating an abstract model of the rogue code insertion process into a targeted wireless communications system that used RF atmospheric signal transmission.

The model was general enough to apply it to widely used target environments such as the UNIX, Macintosh and DOS operating systems. In this experiment, the model was instantiated on a DOS-based system that used a Local Area Wireless Network (LAWN) connection.

This experiment to instantiate the abstract model in chapter 3 tc insert rogue code into a targeted host was successful. The author used the ISO 8473 Connectionless-mode Network Protocol[132] (CLNP) for network-level connectivity, and the Trivial File Transfer Protocol[133] (TFTP), as its basic transmission protocol through applications level connectivity to insert the rogue program. Three IBM PC-compatible computer systems were connected by RF LAN, using O'Neill Communications' Local Area Wireless Network (LAWN) modules. The author copied a file between the two legitimate computer systems to ensure that the file transfer software worked. The third spurious computer system, the imposter, then inserted rogue code into the data stream as the file was being transferred a second time between the same two, legitimate computer systems. The innocent recipient executed the rogue code when it executed the infected file, thereby illustrating the rogue code's successful insertion. Two defense measures, CRCs and checksumming, to prevent rogue code insertions via RF were examined.

The technique to instantiate the abstract model, using specific protocols (CLNP and TFTP) and O'Neill's (UWN) wireless communication modules, may be generalized. The principles and the technique used remain valid for other protocols and other communication modules. Inserting rogue programs can be more complex and sometimes near impossible with current technology, but with unlimited time and resources, it can be done.

Chapter 5 CONTRIBUTIONS, CONCLUSIONS AND IMPLICATIONS FOR FUTURE RESEARCH

5.1 Contributions

Wireless LANs are becoming more and more popular. This popularity increases the opportunities for intruders to infect computing systems via RF. The hardware and software telecommunication components, the specifications for each component and the technology to inject rogue programs via RF communication channels are proven and readily available. Unauthorized users can purchase "Telecommunication Saturday Night Specials" at many electronics outlet to insert rogue code into a communication channel via RF surreptiously.

This dissertation makes three major theoretical and three proof of concept contributions. The first major theoretical contribution is the development of an abstract model of the rogue code insertion process into a wireless network using RF.

The second major theoretical contribution is the development of the methodology and three modules to generate rogue code and insert it into a wireless LAN. The three modules are the prober, activator, and trigger modules.

The third major theoretical contribution is the inclusion of the VL into the abstract model. This was accomplished by combining Fred Cohen's[134] and Linda Rutledge's[135] works with

the proposed topological, vector and functional factors, to establish a computing system's VL to rogue program threats.

The first proof of concept contribution is the finding that inadequately protected wireless LANs are more vulnerable to rogue program attack than traditional LANs. Because of their inherent characteristics, wireless LANs have unique security concerns. They run not only the same risks as traditional LANs, but they also have the additional risks associated with an open transmission medium. Intruders can scan radio waves, and given sufficient time and resources, they can interrupt, analyze, decipher and reinsert data into the communication medium.

The second proof of concept contribution is the demonstration that rogue code could be successfully inserted into a target host via RF. This will not only make the computing community aware of wireless LANs' inherent vulnerabilities, but the insertion will also help the community identify, analyze and neutralize these weaknesses and defend against unauthorized users.

The third proof of concept contribution is the cost-benefit component of the abstract model. The component demonstrated that it generally costs users significantly more not to employ safeguards on their wireless LANs than to employ safeguards.

5.2 Conclusions

The value of this work is that this study can be applied to the UNIX, MVS, Macintosh and other operating systems or other related telecommunication spheres, such as cellular phones, automatic bank tellers, short wave communications, electronic warfare, and satellite manipulation applications.

Cellular phones are popular with all population sectors. By the end of 1994, millions of people will have cellular phones in the USA alone and thousands more will have cellular modems[136]. This technology has provided ample opportunity for

companies to make money and "would be intruders" to cause havoc using the various methods of communicating through the air. Some of these methods include[137]:

1. Cellular Digital Packet Data (CDPD), which is an emerging technology that transmits data over cellular networks by inserting data packets into unused voice channels. Its main use is likely to be for short, bursty transactions, such as mobile credit-card authorizations.
2. Circuit-switched cellular, which uses today's cellular network to transfer connection-oriented data via a cellular modem.
3. Mobile satellite service, which is voice and messaging-oriented technology targeted at places without an existing wired infrastructure.
4. Paging, which is a one-way data messaging and broadcast technology.
5. Enhanced Mobile Radio, which is voice and data technology.

For example[138], UPS uses CDPD-like technology today to the tune of 510,000 to 520,000 calls per day.

Short wave, electronic warfare[139], and satellite manipulation applications are other areas which will become more vulnerable as technology improves. They all function within the radio

frequency spectrum. For example, the FCC recent announcement to allocate thin slices of spectrum in the 2-GHz range to potential service providers may give unauthorized users the opportunity to gain access for illegitimate purposes.

5.3 Future Work

This work points out the need for more research in protocol design. The current protocol suite* use layers to reduce their design complexity and provide well-defined interfaces between the layers, so that a change on one layer doesn't affect an adjacent layer. The protocol suites are[140]:

1. the TCP/IP protocol suite (the Advanced Research Projects Agency (ARPA) Internet protocols),
2. Xerox Network Systems (Xerox NS or XNS),
3. IBM's Systems Network Architecture (SNA),
4. IBM's NetBIOS,
5. the OSI protocols,
6. Unix-to-Unix Copy (UUCP).

* A protocol suite is a collection of protocols from more than one layer that forms the basis of a useful network.

Each of these protocol suites define different protocols at different layers, such as Trivial File Transfer Protocol (TFTP) is one specific user process whose protocol is defined by the TCP/IP protocol suite.

Protocols provide resource sharing and interconnection; security was not a major factor. These protocols generally do not have duplicate packet checking, resulting in discarding any duplicate packets. The rogue code exploited this protocol characteristic in chapter 3 to insert the rogue code into the communication stream of a targeted host via RF. More robust protocols would minimize the threats delineated in this dissertation.

Another extension of this dissertation for future researchers is to conduct an empirical analysis of the Accessibility Vulnerability Likelihood. This dissertation only discussed the variability of VL as n doubles with the other parameters remaining the same. As expected the normalized value of VL remains the same. Further study is needed to determine the effects of VL as all parameters vary to provide a comprehensive perspective of how accessible networks are to rogue programs.

APPENDIX 1 - INITIALISATION CODE

Initialisation Code for Host Aaron (Start.Bat)

```
PATH=a:\;b:\;b:\clnptsr;b:\clnpmgr;a:\packet
SET HOSTNAME=aaron
prompt $t$h$h$h $p$g
a:
cd \packet
call llawn
b:
cd \clnptsr
clnptsr
cd \clnpmgr
```

Initialisation Code for Host Bob (Start.Bat)

```
PATH=a:\;b:\;b:\clnptsr;b:\clnpmgr;a:\packet
SET HOSTNAME=bob
prompt $t$h$h$h $p$g
a:
cd \packet
call llawn
b:
cd \clnptsr
clnptsr
cd \clnpmgr
```

Initialisation Code for Host Intruder (Start.Bat)

```
PATH=a:\;b:\;b:\clnptsr;b:\clnpmgr;a:\packet
SET HOSTNAME=intruder
prompt $t$h$h$h $p$g
a:
cd \packet
call llawn
b:
cd \clnptsr
clnptsr
cd \clnpmgr
```

APPENDIX 2 - ROGUE PROGRAM CODE

```
.Text segment byte public "code"
.Text ends
Assume CS: .Text
Text      segment

Label     msgbegin near
          db "This program has been infected by a harmless
virus",0
Label     msgsend near

Virus     proc  near
          mov bx,2
          mov cx,offset ( msgsend - msgbegin )
          mov dx,cs
          mov ds,dx
          mov dx, offset msgbegin
          mov ah, 48H
          int 21H
          mov ah,4cH
          int 21H

Virus     enddp
.Text   ends
.Data   segment
.Data   end

End     Virus
```

Endnotes

1. Dobbins, J.H., "Software Acquisition Management," Program Manager, Journal of the Defense Systems Management College, January-February 1994, pages 2-8.

2. Hoffman, Lance J., "Rogue Programs: Viruses, Worms, and Trojan Horses," Van Nostrand Reinhold, 115 Fifth Avenue, New York, N.Y. 10003, 1990, page xi.

3. Denning, Peter, "Computers Under Attack - Intruders, Worms, and Viruses", ACM Press, New York, N.Y. 1990, page xiii.

4. USENIX, The Journal of the USENIX Association, University of California Press, Vol. 2, Spring 1989, pages 155-176.

5. Computers & Security, Elsevier Advanced Technology, Volume 9, number 5, August 1990.

6. Markoff, J., "U.S. Is Moving to Restrict Access to Facts About Computer Virus," New York Times, Nov 11, 1988.

7. Slade, Robert, "Antivirus Contact List," Vancouver Institute for Research into User Security, Integrity Canada, V7K2G6, Internet address is Robert_Slade@mtsg.sfu.ca, 1991.

8. Highland, H., "The Brain Virus: Fact and Fantasy," Computers and Security, Vol. 7, August 1988, page 367.

9. Denning, Peter, "Computers Under Attack - Intruders, Worms, and Viruses," ACM Press, New York, N.Y. 1990, page xii.

10. Stoll, Clifford, "Stalking the Wily Hacker," Communications of the ACM, Vol. 31, No. 5, May 1988.

11. Dataquest Research, 600 Delran Parkway, Delran, NJ 08075 1990.

12. Hoffman, P., "VSUMX", Virus Summary List, 333 Bowers Ave, Suite 130, Santa Clara, CA., Dec 1993.

13. NCR Corporation, NCR WAVELAN, Dayton, Ohio, 1990, telephone number: 1-800-225-5627.

14. Perry, D., Business Communications Review, "Will Wireless LANs Realize Their Potential?", Vol 23, Issue 8, Aug 1993, page 19.

15. Lathrop, D., "Security Aspects of Wireless LANs," Computers and Security, Vol. 11, 1992, pages 421-426.

16. Eichin, Mark W., and Rochlis, Jon A., "With Microscope and Tweezers: The Worm from MIT's Perspective," Communications of the ACM, Vol. 32, No.6, June 1989.

17. Seeley, Donn, "A Tour of the Worm," The Computer Worm - A Report to the Provost of Cornell University, Cornell University, Ithaca, N.Y. 14853, Feb 1989.

18. Spafford, Eugene, H., "The Internet Worm: Crisis and Aftermath," Department of Computer Sciences, Purdue University, West Lafayette, IN., 47907.

19. United States General Accounting Office, "Computer Security," June 1989, GAO/IMTEC-89-57.

20. Trusted Information Systems, Inc., "Computer System Intrusion Detection," Contract No. F30602-87-D-0093, Sept 11, 1990, page 8.

21. Trusted Information Systems, Inc., "Computer System Intrusion Detection," Contract No. F30602-87-D-0093, Sept 11, 1990.

22. The Computer Security Alert, "Distributed Intrusion Detection System," Computer Security Institute, San Francisco, CA., 94107, No. 107, Feb 1992, pages 3-8.

23. Trusted Information Systems, Inc., "Computer System Intrusion Detection," Contract No. F30602-87-D-0093, Sept 11, 1990, pages 30-35 (Appendix B).

24. Ibid, pages 17-23.

25. Ibid, pages 1-5.

26. Ibid, pages 11-16.

27. Ibid, pages 24-29.

28. Ibid, pages 36-44.

29. Ibid, pages 45-47.

30. Ibid, pages 48-53.

31. The Computer Security Alert, "Distributed Intrusion Detection System," Computer Security Institute, San Francisco, CA., 94107, No. 107, Feb 1992, pages 3-8.

32. Fischhoff, B., "The Art of Cost-Benefit Analysis", Defense Technical Information Center, Cameron Station, VA, Feb 1984, page 2-1.

33. Cassady, P., "Integrated Family of Test Equipment Electro-Optical Program Cost-Benefit Analysis", Defense Technical Information Center, Cameron Station, VA, Dec 1990, pages 9-12.

34. Cohen, F., "A Cost Analysis of Virus Defenses," A Short Course on Computer Viruses, ASP Press, PO Box 81270, Pittsburgh, PA., 1990, pages 155-160.

35. Cohen, F., "A Note on the use of Pattern Matching in Computer Virus Detection", Invited Paper, Computer Security Conference, London, England Oct 11-13, 1989.

36. Hirst, J., "Eliminator - Virus Detection and Removal", Users Manual, British Computer Virus Research Center, 1990.

37. Cohen, F., "A Cryptographic Checksum for Integrity Protection", Computers and Security, Vol. 6, No. 6, 1987, pages 505-510.

38. Cohen, F., "Models of Practical Defenses Against Computer Viruses", Computers and Security, Vol. 7, No. 6, 1988, pages 308-313.

39. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, Mar 1987, page 97.

40. Fischhoff, B., "The Art of Cost-Benefit Analysis," Decision Research, 1201 Oak St., Eugene, Oregon, Jul 1978, pages 1-1 to 4-14.

41. Burger, R., "Computer Viruses - A High Tech Disease," Abacus, 52nd Street SE, Grand Rapids, MI., 49508, page 13.

42. Microsoft, "MS-DOS Programmer's Reference," Microsoft Press, Redmond WA., 98052, 1991, pages 115, 116.

43. Burger, R., "Computer Viruses - A High Tech Disease," Abacus, Grand Rapids, MI., 49508, 1988, page 98.

44. Ibid, pages 98-99.

45. Hoffman, P., "Virus Information Summary List", 3333 Bowers Avenue, Santa Clara, CA., 95054, Dec 1993, page 5.

46. Burger, R., "Computer Viruses - A High Tech Disease," Abacus, Grand Rapids, MI., 49508, 1988, page 100.

47. Dettman, T., DOS Programmer's Reference, Que Co., 11711 N.College Ave., Carmel, IN., 46032, 1989, page 250.

48. Tanenbaum, A., "Computer Networks", Prentice-Hall, Inc., Englewood Cliffs, NJ., 07632, 1981, page 2.

49. Pfleeger, C., "Security in Computing," Prentice-Hall, Englewood Cliffs, NJ, 1989, page 365.

50. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., 1989, page 403.

51. Tanenbaum, A., "Computer Networks," Prentice-Hall, 1988, pages 117, 118.

52. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., 1989, page 371.

53. Ibid.

54. PIN, Warfel and Miller Inc., Vol. 6, No. 6, July 1990, page 4.

55. National Institute of Standards and Technology (NIST) Special Publication 500-157, "Smart Card Technology: New Methods for Computer Access Control," Computer Science and Technology, US Department of Commerce, Sept 1988.

56. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., 1989, pages 258-269.

57. Ames, S., "Security Kernel Design and Implementation: an Introduction," Computer, Vol. 16, No. 7, Jul 83, pages 14-23.

58. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., 1989, page 276.

59. NCR Corporation, NCR WAVELAN, Dayton, Ohio, telephone number: 1-800-225-5627, 1990.

60. Altair Product Operations, 108 East 91st Street, New York, NY., 10128, telephone number: 1-800-233-0877.

61. O'Neill Communications, Inc., "The LAWN," Thanet Circle, Princeton, NJ., 08540, telephone number: 1-609-497-6800.

62. Proxim, Inc., "ProxNet," Mountain View, CA., telephone number: 1-415-960-1630.

63. Telesystems SLW Inc., "Arlan," Don Mills, Ontario.

64. California Microwave Inc., "Radio Link," 985 Almanor Ave, Sunnyvale, CA., 94086, telephone number: 1-800-772-5465.

65. The Black Box Corporation, "BestLAN," Lancaster, PA., telephone number: 1-412-746-5565.

66. IBM, Marketing Division for Wireless LANs, telephone number: 1-800-426-3333.

67. Kramer, M., "Infrared Schemes Offer Alternative to Radio LANS," PC Week Magazine, June 3, 1991, page 93.

68. Tanenbaum, A., Computer Networks, Prentice-Hall, Inc., Englewood Cliffs, NJ., 07632, 1981, page 275.

69. Proc. APIPS NCC, 1975, pages 203-215.

70. Broadcasting, "Captain Midnight Strikes; Preempts HBO with Message Decrying Scrambling," Washington D.C., July 28, 1986, page 71.

71. USENIX, The Journal of the USENIX Association, University of California Press, Vol. 2, Spring 1989, pages 155-176.

72. Burger, R., "Computer Viruses: A High Tech Disease," Abacus, Grand Rapids, MI., 1989.

73. Communications of the ACM, Vol. 32, No. 6, June 1989.

74. ABACUS, Vol. 4, No. 4, Summer 1987.

75. Ludwig, M., "The Little Black Book of Computer Viruses", American Eagle Publications, Inc., Post Office Box 41401, Tucson, Arizona 85717, 1991.

76. Defense System Management College, "Software Acquisition Strategies Caselet", Software Management, Ft. Belvoir, VA., July 1993, page 4.

77. Ibid.

78. Computergram International, Technology News of America Co., 110 Green St., Rm. 1101, New York, NY., 10012, Feb 21, 1991, No. 1617, page 3.

79. IEEE Network, Vol. 1, No. 5, Nov 1990, page 10.

80. Seybold, Patricia, "Network Monitor," Version 6, No. 6, Jun 1991, page 24.

81. Computerworld, "Software Distribution Key to Open Systems," Vol. 27, No. 42, Oct 18, 1993, page 85.

82. Computer Protection Systems, Inc. "LAN Security," 150 N. Main, Plymouth, Michigan 48170, Vol. X, No. 10, Jan 1993, page 3.

83. Cohen, F., "Computer Viruses - Theory and Experiments," Computers and Security, Vol. 6, No. 1, 1987, pages 22-35.

84. Cramer, Myron, and Pratt, Stephen, "Computer Virus Countermeasures- A New Type of Electronic Warfare," Defense Electronics, Oct 1989, pages 75-84.

85. Cramer, M., and Pratt, S., "Computer Virus Countermeasures - A New Type of Electronic Warfare," Rogue Programs: Viruses, Worms, and Trojan Horses, edited by Lance J. Hoffman, Van Nostrand Reinhold, New York, NY., 1990, pages 246-260.

86. GAO, "Virus Highlights Need for Improved Internet Management," GAO/IMTEC-89-57, June 1989, page 39.

87. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., Englewood Cliffs, NJ., 1989, pages 366-369.

88. Hoffman, P., "VSUMX", Virus Summary List, 333 Bowers Ave, Suite 130, Santa Clara, CA., Aug 1993, page 10.

89. Tanenbaum, A., "Computer Networks," Prentice-Hall, NJ., 1981, page 129.

90. Tanenbaum, A., "Computer Networks", Prentice-Hall, NJ., 1981, page 130.

91. Hoffman, L., "Rogue Programs: Viruses, Worms, and Trojan Horses," Van Nostrand Reinhold, NY., 1990, page 19.

92. Mayo, J., "Computer Viruses," Windcrest Books, Blue Ridge Summit, PA., 1989, page 106.

93. Ferbrache, D., "A Pathology of Computer Viruses," Springer-Verlag London, 1992, pages 110-111.

94. Schneider, B., "Making Sense of Encryption," Infosecurity News, Vol. 4, No. 2, March/April 1993, page 37.

95. Pozzo, Maria and Gray, Terence, E., "An Approach to Containing Computer Viruses", Computers and Security, Vol. 6, 1987, page 17.

96. Pfleeger, C., "Security in Computing," Prentice-Hall, Englewood Cliffs, NJ., 07632, 1989, page 25.

97. Ibid.

98. Ibid.

99. Parker, S., Encyclopedia of Electronics and Computers, McGraw-Hill Inc., 1988, page 180.

100. U.S. Department of Commerce, "Digital Signature Standard", Computer Systems Laboratory (CSL) Bulletin, National Institute of Standards and Technology, January 1993.

101. Feudo, C., "The Computer Virus Desk Reference," Business One Irwin, Homewood, IL., 60430, 1992, pages 105-107.

102. Hoffman, Lance J., "Computer Viruses: A Plea for Sanity," presented at the Invitational Workshop on Computer Viruses, Oct 1988, New York, N Y., page 1.

103. Pfleeger, C.P., "Security in Computing," Prentice-Hall, Englewood Cliffs, New Jersey, 1989, pages 462-463.

104. Gardner, P., "Five Risk Assessment Programs," Computers and Security, Vol. 8, No. 6, Oct 1989, pages 291-296.

105. Cohen, F., "A Cost Analysis of Virus Defenses," A Short Course on Computer Viruses, ASP Press, PO Box 81270, Pittsburgh, PA., 1990, pages 155-160.

106. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, March 1987, page 97.

107. Cohen, F., "A Cost Analysis of Virus Defenses," A Short Course on Computer Viruses, ASP Press, PO Box 81270, Pittsburgh, PA., 1990, pages 155-160.

108. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, March 1987, page 97.

109. Cohen, F., "A Cost Analysis of Typical Computer Viruses and Defenses", ASP Press, PO Box 81270, Pittsburgh, PA., 1990.

110. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, March 1987, page 97.

111. International Organization of Standards, ISO 8473 1988, "Information Processing Systems - Data Communications - Protocol for Providing the Connectionless-Mode Network Service".

112. Sollins, K., The TFTP Protocol, Network Working Group, Request for Comments: 783, MIT, June 1981.

113. Tanenbaum, A., "Computer Networks", second edition, Prentice-Hall Inc., 1988, pages 358, 429-431.

114. Tanenbaum, A., "Computer Networks", second edition, Prentice-Hall Inc., 1988, pages 358-361.

115. International Organization of Standards, ISO 8473 1988, "Information Processing Systems - Data Communications - Protocol for Providing the Connectionless-Mode Network Service".

116. Sollins, K., The TFTP Protocol, Network Working Group, Request for Comments: 783, MIT, June 1981.

117. Pfleeger, C., "Security in Computing," Prentice-Hall, Inc., NJ., 1989, page 366.

118. International Organization of Standards, ISO 8473 1988, page 1.

119. Sollins, K., The TFTP Protocol, Network Working Group, Request for Comments: 783, MIT, June 1981.

120. Stevens, R., "UNIX Network Programming," Prentice-Hall Software Series, 1990, page 465.

121. Comer, D., "Internet Working with TCP/IP: Principles, Protocols, and Architecture," Prentice-Hall Inc., 1988, page 239.

122. International Organization of Standards, ISO 8473 1988, page 14.

123. International Organization of Standards, ISO 8473 1988, "Information Processing Systems - Data Communications - Protocol for Providing the Connectionless-Mode Network Service", pages 6, 24.

124. Pickholtz, D., Schilling, D., and Milstein, L., "Theory of Spread-Spectrum Communications - A Tutorial," GWU-IIST-81-31, May 1982, pages 855-884.

125. Pickholtz, D., Schilling, D., and Milstein, L., "Theory of Spread-Spectrum Communications - A Tutorial," GWU-IIST-81-31, May 1982, pages 855-884.

126. Pickholtz, R., et al, "Spread Spectrum Goes Commercial," IEEE Spectrum, Aug 1990, page 40.

127. International Organization of Standards, ISO 8473 1988, "Information Processing Systems - Data Communications - Protocol for Providing the Connectionless-Mode Network Service".

128. Nelson, M., Dr. Dobb's Journal, "File Verification Using CRC," Vol. 17, No. 5, May 1992, pages 61-68.

129. Dettmann, T., "DOS Programmer's Reference," Que Corporation, 11711 N. College Ave., Carmel, IN., 46032, 1989, pages 262-263.

130. Cohen, F., "A Cost Analysis of Virus Defenses," A Short Course on Computer Viruses, ASP Press, PO Box 81270, Pittsburgh, PA., 1990, pages 155-160.

131. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, March 1987, page 97.

132. International Organization of Standards, ISO 8473 1988, "Information Processing Systems - Data Communications - Protocol for Providing the Connectionless-Mode Network Service".

133. Sollins, K., The TFTP Protocol, Network Working Group. Request for Comments: 783, MIT, June 1981.

134. Cohen, F., "A Cost Analysis of Virus Defenses," A Short Course on Computer Viruses, ASP Press, PO Box 81270, Pittsburgh, PA., 1990, pages 155-160.

135. Rutledge, L., "A Spatial Encoding Mechanism For Network Security," The George Washington University, GWU-IIST-87-04, March 1987, page 97.

136. Computerworld, "Enterprising Networks," Oct 11, 1993, Vol. 27, No. 41, pages 51-54.

137. Ibid, page 51.

138. Ibid, page 54.

139. Evancce, P., and Bentley, M., "Computer Viruses Loom as Future Era Weapons", Defense Journal, February 1994, pages 19-21.

140. Stevens, R., "UNIX Network Programming", Prentice Hall, Englewood Cliffs, NJ, 1990, pages 171-196.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT "A" distribution for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Defense Systems Management College TR 4-94	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Defense Sys Mgmt College 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Same as 6a.
6c. ADDRESS (City, State, and ZIP Code) 9820 Belvoir Rd Ste G38 Ft. Belvoir, VA 22060-5565	7b. ADDRESS (City, State, and ZIP Code) Same as 6c.
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

An Abstract Model of Rogue Code Insertion into Radio Frequency Wireless Networks

12. PERSONAL AUTHOR(S)
LtC Christopher V. Feudo

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical Report	FROM _____ TO _____		

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A dissertaion presented to the faculty of the School of Engineering and Computer Science, the George Washington University. This research examines the effects of computer viruses to the Program Management Office.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Sylvia Nance	22b. TELEPHONE (Include Area Code) (703) 805-2376 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE